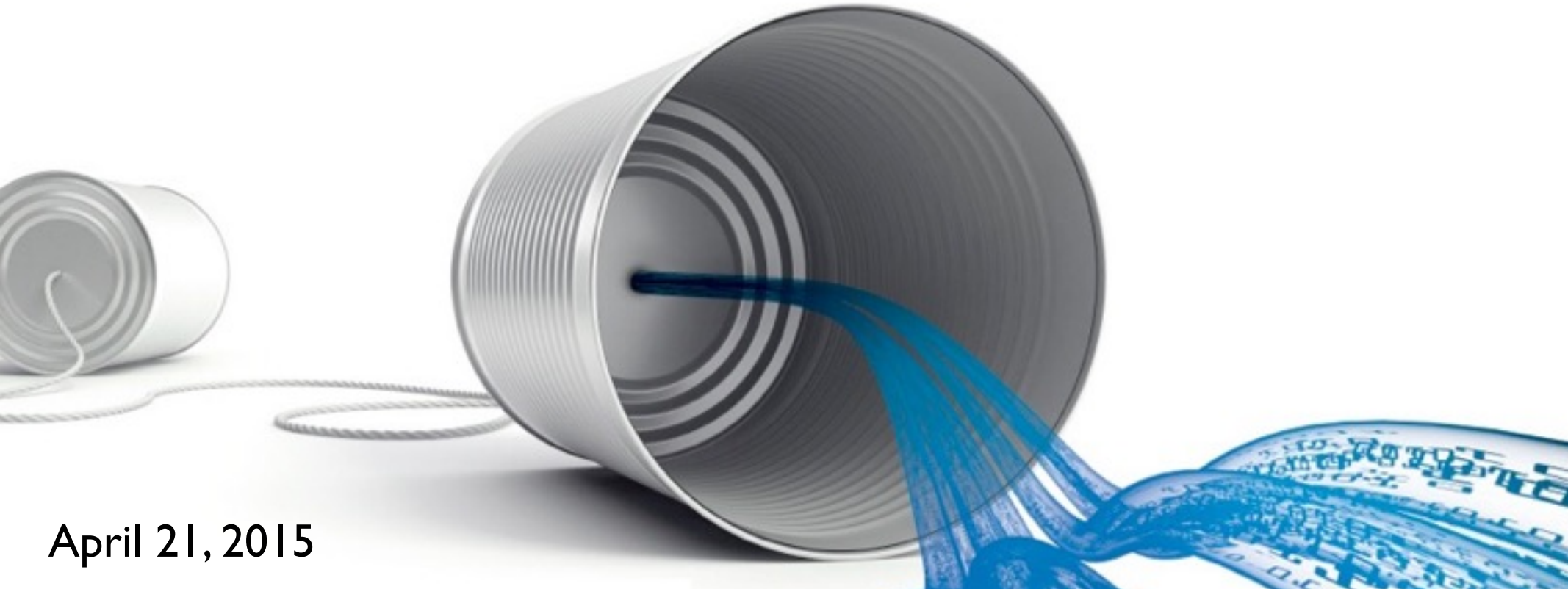
Great Theoretical Ideas in Computer Science

Communication Complexity



April 21, 2015

What are the limitations to what computers can learn?

Do certain mathematical theorems have short proofs?

Can quantum mechanics be exploited to speed up computation?

Is every problem whose solution is efficiently verifiable also efficiently solvable? *i*.e. P = NP?

What are the limitations to what computers can learn?

Do certain mathematical theorems have short proofs?

Can quantum mechanics be exploited to speed up computation?

Is every problem whose solution is efficiently verifiable also efficiently solvable? *i.e.* P = NP?

Communication complexity

Cool Things About Communication Complexity

Many useful applications:

machine learning, proof complexity, quantum computation, pseudorandom generators, data structures, game theory,…

The setting is simple and neat.

Beautiful mathematics

combinatorics, algebra, analysis, information theory

Motivating Example 1: Checking Equality

0100101011110101 $\stackrel{?}{=}$ 0100101001110101

$\longleftarrow n$ bits $\longrightarrow$ $\longleftarrow n$ bits $\longrightarrow$

How many bits need to be communicated?

Naively: n Actually: n

What if we allow 0.00000000001% probability of error?

Naively: $\Omega(n)$ Actually: $O(\log n)$

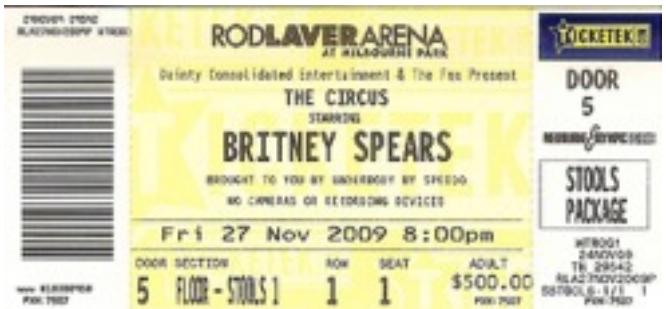
Motivating Example 2: Auctions

Alice

$100

Bob

$1000



Defining the model a bit more formally

2 Player Model of Communication Complexity

$$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

(Alice)

$x \in \{0,1\}^n$

(Bob)

$y \in \{0,1\}^n$

Goal: Compute $F(x,y)$. (both players should know the value)

How: Sending bits back and forth according to a protocol.

Resource: Number of communicated bits.

2 Player Model of Communication Complexity

$$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

(Alice)

$x \in \{0,1\}^n$

(Bob)

$y \in \{0,1\}^n$

cost of protocol = $\max_{(x,y)}$ # bits communicated for (x, y).

Deterministic communication complexity

$\mathbf{D}_2(F) =$ min cost of a protocol computing F.

Randomized communication complexity

$\mathbf{R}_2^{\epsilon}(F) =$ min cost of a randomized protocol computing F.

2 Player Model of Communication Complexity

$$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

(Alice)

$x \in \{0,1\}^n$

(Bob)

$y \in \{0,1\}^n$

$$0 \leq \quad \mathbf{R}_2^{\epsilon}(F) \leq \mathbf{D}_2(F) \quad \leq n+1$$

$$c \qquad \log^c(n) \qquad\qquad n^{\delta} \qquad \delta n$$

Examples

Equality: $$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{D}_2(EQ) = n + 1. \qquad \mathbf{R}_2^{\epsilon}(EQ) = O(\log n).$$

Parity: $$PAR(x,y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$

$$\mathbf{D}_2(PAR) = O(1)$$

Examples

Can view the input string as a subset of {1,2,3,…,n}

$$S_x = \{2, 4, 5\}$$

$$x = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline \end{array}$$

1 2 3 4 5 6 7 8

Average: $AVG(S_x, S_y) = \text{average of multiset } S_x \cup S_y$

Note: the range of the function is not {0, 1}

$$\mathbf{D}_2(AVG) = O(\log n)$$

(Alice sends $\sum_{i \in S_x} i$ and $|S_x|$)

Median: $MED(S_x, S_y) = \text{median of multiset } S_x \cup S_y$

$$\mathbf{D}_2(MED) = O(\log^2 n)$$ homework?

Examples

Can view the input string as a subset of {1,2,3,…,n}

$$S_x = \{2, 4, 5\}$$

$$x = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline \end{array}$$

1 2 3 4 5 6 7 8

Disjointness: $DISJ(S_x, S_y) = \begin{cases} 1 & S_x \cap S_y = \emptyset \\ 0 & \text{otherwise} \end{cases}$

$\mathbf{R}_2^{\epsilon}(DISJ) = \Omega(n).$ hard

The plan

1. Efficient randomized communication protocol for checking equality.

2. Two applications of communication complexity.

3. How to prove lower bounds.

Efficient randomized communication protocol for checking equality

The Power of Randomization

$$\mathbf{R}_2^{\epsilon}(EQ) = O(\log n).$$

The Protocol:

Alice's Input: $a_0 a_1 a_2 \ldots a_{n-1} \in \{0,1\}^n$

Bob's Input: $b_0 b_1 b_2 \ldots b_{n-1} \in \{0,1\}^n$

Alice picks a prime $p \in [n^2, 2n^2]$ and a random $t \in \mathbb{Z}_p$.

Alice builds polynomial

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \in \mathbb{Z}_p[x]$$

Alice sends Bob: $p,\ t,\ A(t) \rightarrow O(\log n)$ bits

The Power of Randomization

$$\mathbf{R}_2^{\epsilon}(EQ) = O(\log n).$$

The Protocol:

Alice picks a prime $p \in [n^2, 2n^2]$ and a random $t \in \mathbb{Z}_p$.

Alice builds polynomial

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \in \mathbb{Z}_p[x]$$

Alice sends Bob: $p,\ t,\ A(t) \rightarrow O(\log n)$ bits

Bob builds polynomial $B(x) \in \mathbb{Z}_p[x]$

Output: If $A(t) = B(t)$, output *1*. Otherwise, output *0*.

The Power of Randomization

$$\mathbf{R}_2^{\epsilon}(EQ) = O(\log n).$$

Analysis:

Want to show: For all inputs (a, b), probability of error is $\leq \epsilon$.

For all (a, b) with $a = b$:

$$\Pr_t[\text{error}] = \Pr_t[A(t) \neq B(t)] = 0$$

For all (a, b) with $a \neq b$:

$$\Pr_t[\text{error}] = \Pr_t[A(t) = B(t)] = \Pr_t[(A - B)(t) = 0]$$

$$= \Pr_t[t \text{ is a root of } A - B] \leq \frac{n-1}{p} \leq \frac{n-1}{n^2} \leq \frac{1}{n}$$

$\text{degree}(A - B) \leq n - 1$

Two applications of communication complexity

Applications of Communication Complexity

- circuit complexity
- time/space tradeoffs for Turing Machines
- VLSI chips
- machine learning
- game theory
- data structures
- proof complexity
- pseudorandom generators
- pseudorandomness
- branching programs
- data streaming algorithms
- quantum computation
- lower bounds for polytopes representing NP-complete problems

Applications of Communication Complexity

- circuit complexity
- **time/space tradeoffs for Turing Machines**
- VLSI chips
- machine learning
- game theory
- data structures
- proof complexity
- pseudorandom generators
- pseudorandomness
- branching programs
- **data streaming algorithms**
- quantum computation
- lower bounds for polytopes representing NP-complete problems

How Communication Complexity Comes In

Setting: Solve some task while minimizing some resource.

e.g. design a small circuit, find a short proof of a theorem, find a fast algorithm

Goal: Prove lower bounds on the resource needed.

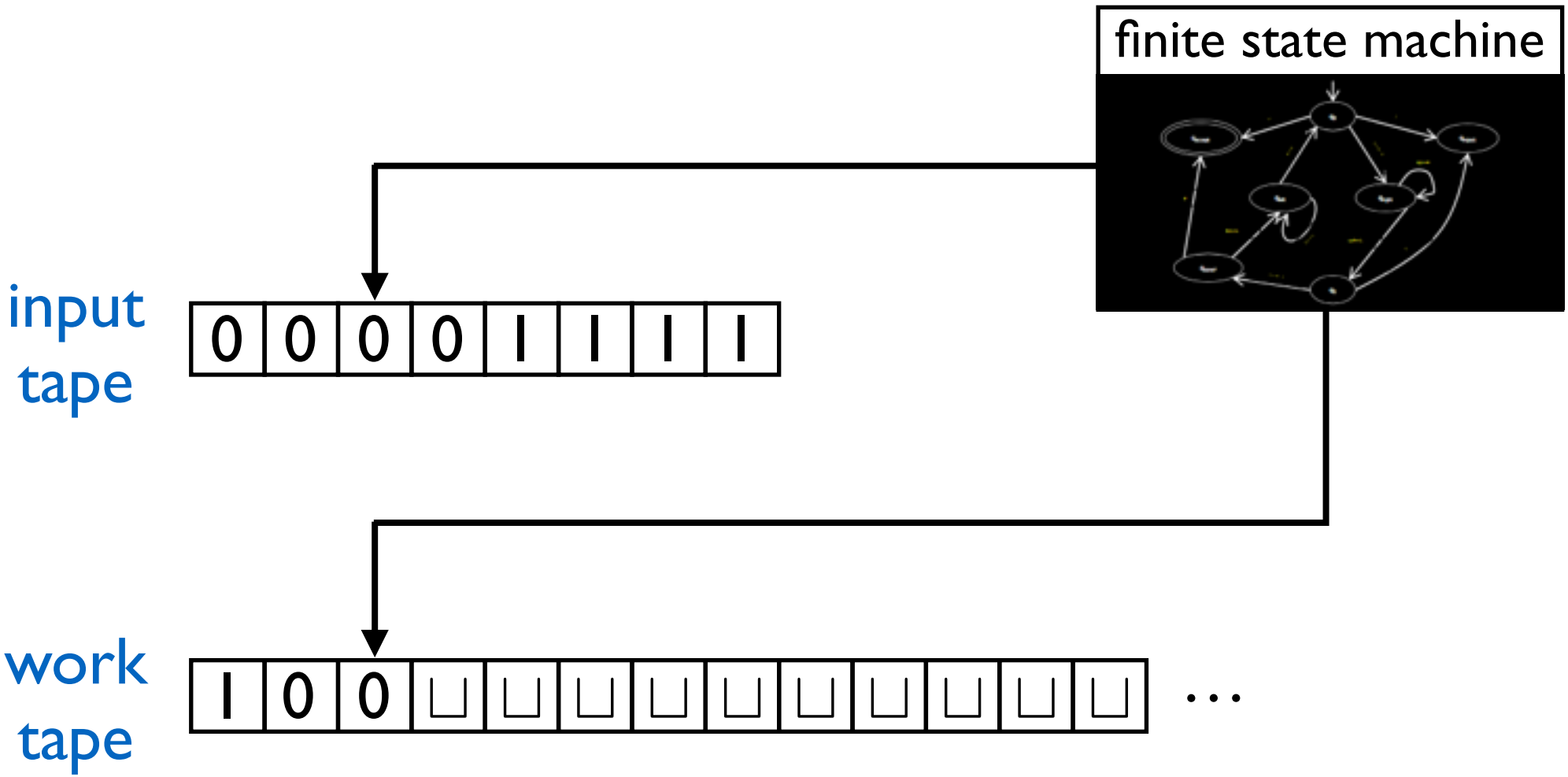
Sometimes: efficient solution to our problem implies efficient communication protocol computing a certain function.

i.e. no efficient protocol for the function implies no efficient solution to our problem.

Many optimization problems contain an implicit *communication bottleneck*.

Time/space tradeoffs for TMs

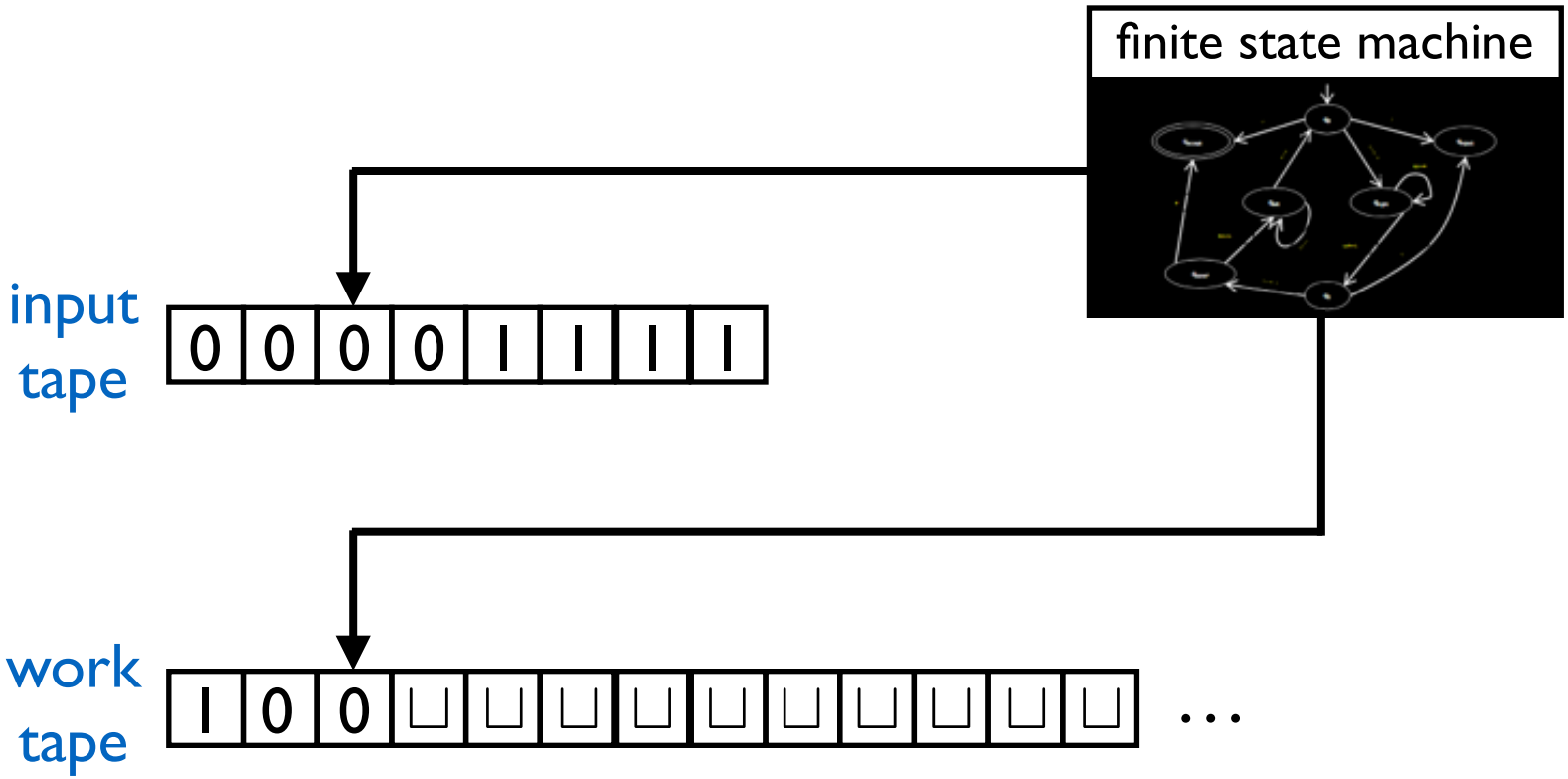
Recall Turing Machines



$T(n)$ time: # steps the machine takes

$S(n)$ space: # work tape cells the machine uses

An observation



Suppose we both know the input and the TM.

You start running the TM with the input.

You pause after a certain number of steps.

What information do I need to be able to continue the computation from where you left it?

1. current state
2. positions of tape heads
3. contents of work tape

Time/space tradeoff for a simple language

Let $L = \{x\#^{|x|}x : x \in \{0,1\}^*\}$

$000\#\#\#000 \in L$

$1010\#\#\#\#1010 \in L$

$001\#\#\#000 \notin L$

$000\#\#000 \notin L$

Theorem:

If a (2-tape) TM decides L in $T(n)$ time and $S(n)$ space on inputs of size $3n$, then $T(n) \cdot S(n) = \Omega(n^2)$.

Time/space tradeoff for a simple language

Let $L = \{x\#^{|x|}x : x \in \{0,1\}^*\}$

Theorem:

If a (2-tape) TM decides L in $T(n)$ time and $S(n)$ space on inputs of size $3n$, then $T(n) \cdot S(n) = \Omega(n^2)$.

Strategy:

Using this TM, we design a communication protocol for *EQ* of cost $O(T(n)S(n)/n)$.

We know *EQ* requires at least $n+1$ bits of comm.

$$T(n)S(n)/n = \Omega(n) \implies T(n)S(n) = \Omega(n^2)$$

Time/space tradeoff for a simple language

Let $L = \{x\#^{|x|}x : x \in \{0,1\}^*\}$

Let M be a TM deciding L.

Protocol for *EQ* of cost $O(T(n)S(n)/n)$:

Given input $x \in \{0,1\}^n$ to Alice, and $y \in \{0,1\}^n$ to Bob.

Alice runs M on input string $x\#^n?$

Once input tape head reaches the $?$ symbol,

Alice sends

1. current state	$O(1)$
2. position of work tape head	$O(\log S(n))$
3. contents of work tape	$O(S(n))$

Bob continues computation of M,
assuming the input is $?\#^n y$

Time/space tradeoff for a simple language

Given input $x \in \{0,1\}^n$ to Alice, and $y \in \{0,1\}^n$ to Bob.

Alice runs M on input string $x\#^n?$

Once input tape head reaches the $?$ symbol,

Alice sends

1. current state	$O(1)$
2. position of work tape head	$O(\log S(n))$
3. contents of work tape	$O(S(n))$

Bob continues computation of M,
assuming the input is $?\#^n y$

Once input tape head reaches the $?$ symbol,

Bob sends

1. current state	$O(1)$
2. position of work tape head	$O(\log S(n))$
3. contents of work tape	$O(S(n))$

•••

Time/space tradeoff for a simple language

They continue until M halts and gives an answer.

So they compute $M(x\#^n y)$.

The output of M is the output of the protocol.

Correctness: M accepts iff $x = y$.

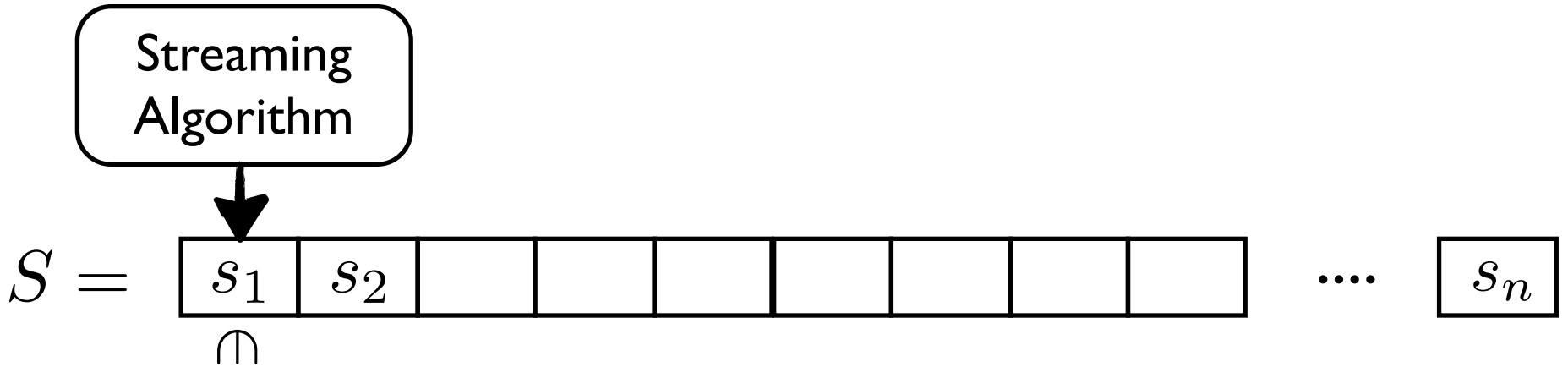
Cost: Each transmission is $O(S(n))$ bits.

tranmissions: $\leq T(n)/n$ (need $\geq$ n steps to reach ? symbol)

Total: $O(T(n)S(n)/n)$

Lower bounds for data streaming algorithms

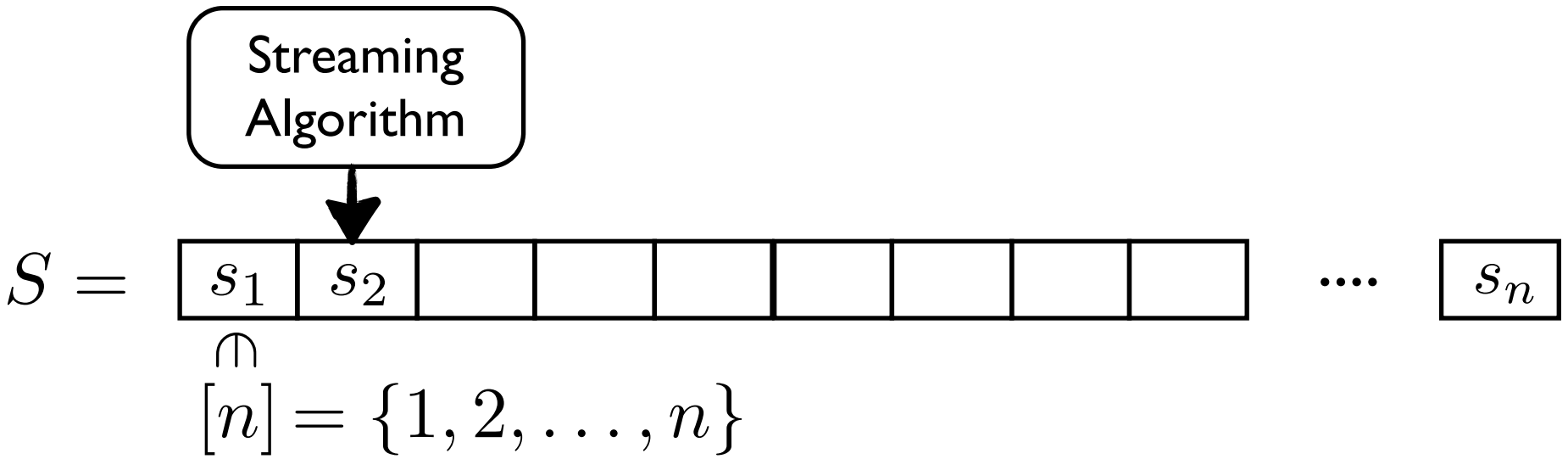
Data Streaming Algorithms



$$[n] = \{1, 2, \ldots, n\}$$

$$S \in [n]^n$$

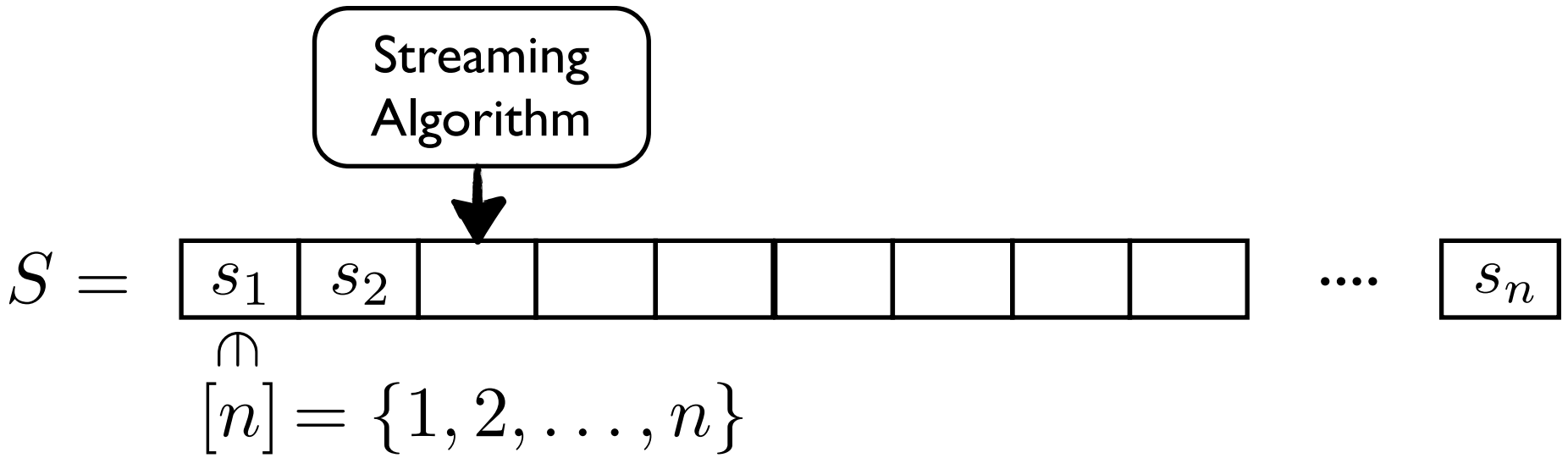
Data Streaming Algorithms



$$[n] = \{1, 2, \ldots, n\}$$

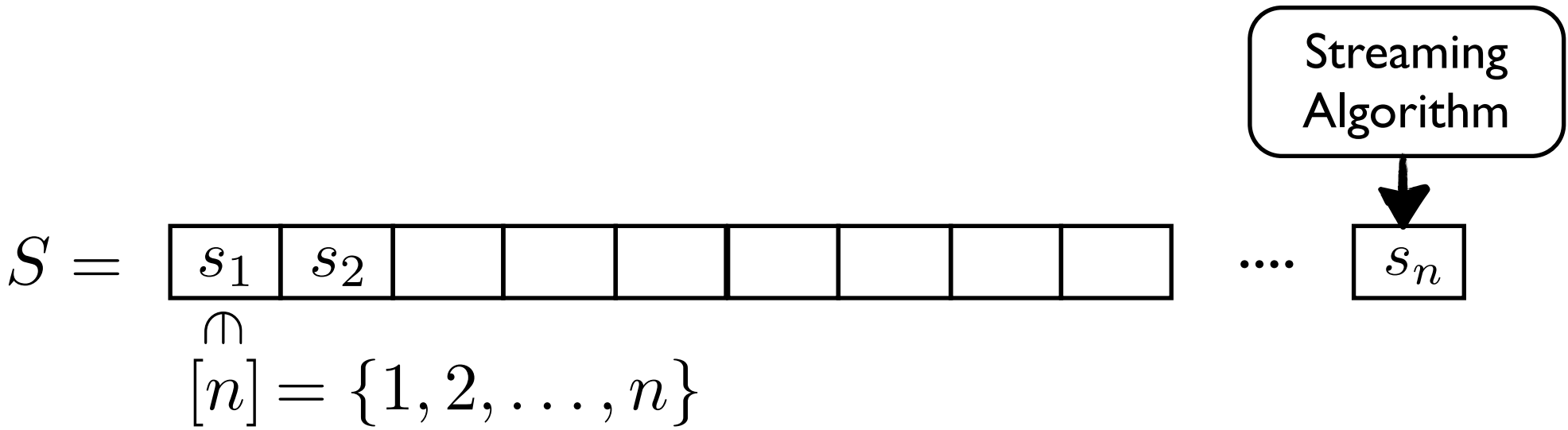
$$S \in [n]^n$$

Data Streaming Algorithms



$S \in [n]^n$

Data Streaming Algorithms



$S \in [n]^n$

Fix some function $f : [n]^n \to \mathbb{Z}$.

e.g. $f(S) = \#$ most frequent symbol in S

Goal: On input S, compute (or approximate) $f(S)$ while minimizing space usage.

Lower Bounds via Communication Complexity

$$f(S) = \#\text{ most frequent symbol in } S$$

Space efficient streaming algorithm computing f 

communication efficient protocol computing $DISJ$.

Disjointness: $DISJ(S_x, S_y) = \begin{cases} 1 & S_x \cap S_y = \emptyset \\ 0 & \text{otherwise} \end{cases}$

Lower Bounds via Communication Complexity

$$f(S) = \#\text{ most frequent symbol in } S$$

Space efficient streaming algorithm computing f ⟶

communication efficient protocol computing $DISJ$.

$S_x = \{2, 4, 5\}$

$x =$

1	2	3	4	5	6	7	8
0	1	0	1	1	0	0	0

$S_y = \{1, 5, 7, 8\}$

$y =$

1	2	3	4	5	6	7	8
1	0	0	0	1	0	1	1

Protocol: Alice runs streaming algorithm on S_x.

She sends the state and memory contents to Bob.

Bob continues to run the algorithm on S_y.

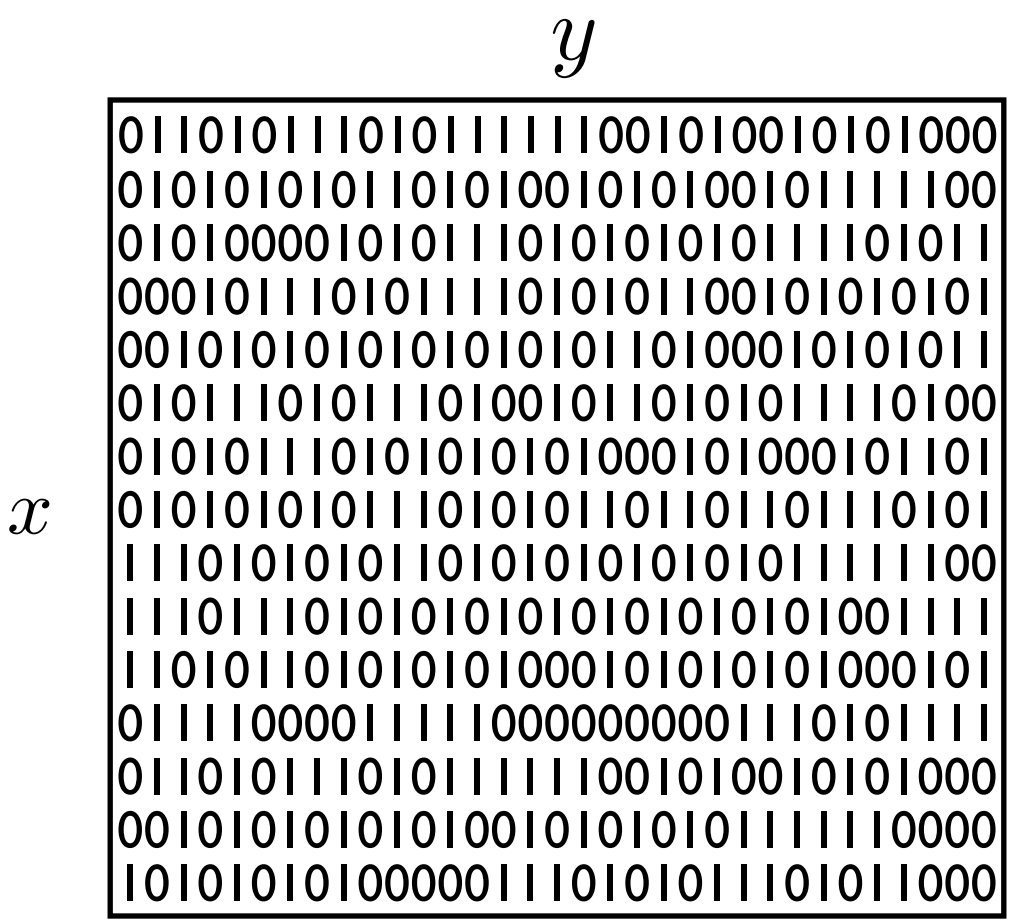
If $f(S_x \cdot S_y) = 2$, Bob outputs 0, otherwise 1.

Correctness ✓ Cost ✓

Lower bounds for deterministic communication complexity

The function matrix

$$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$



$$M[x,y] = F(x,y)$$

The function matrix

Equality: $EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$

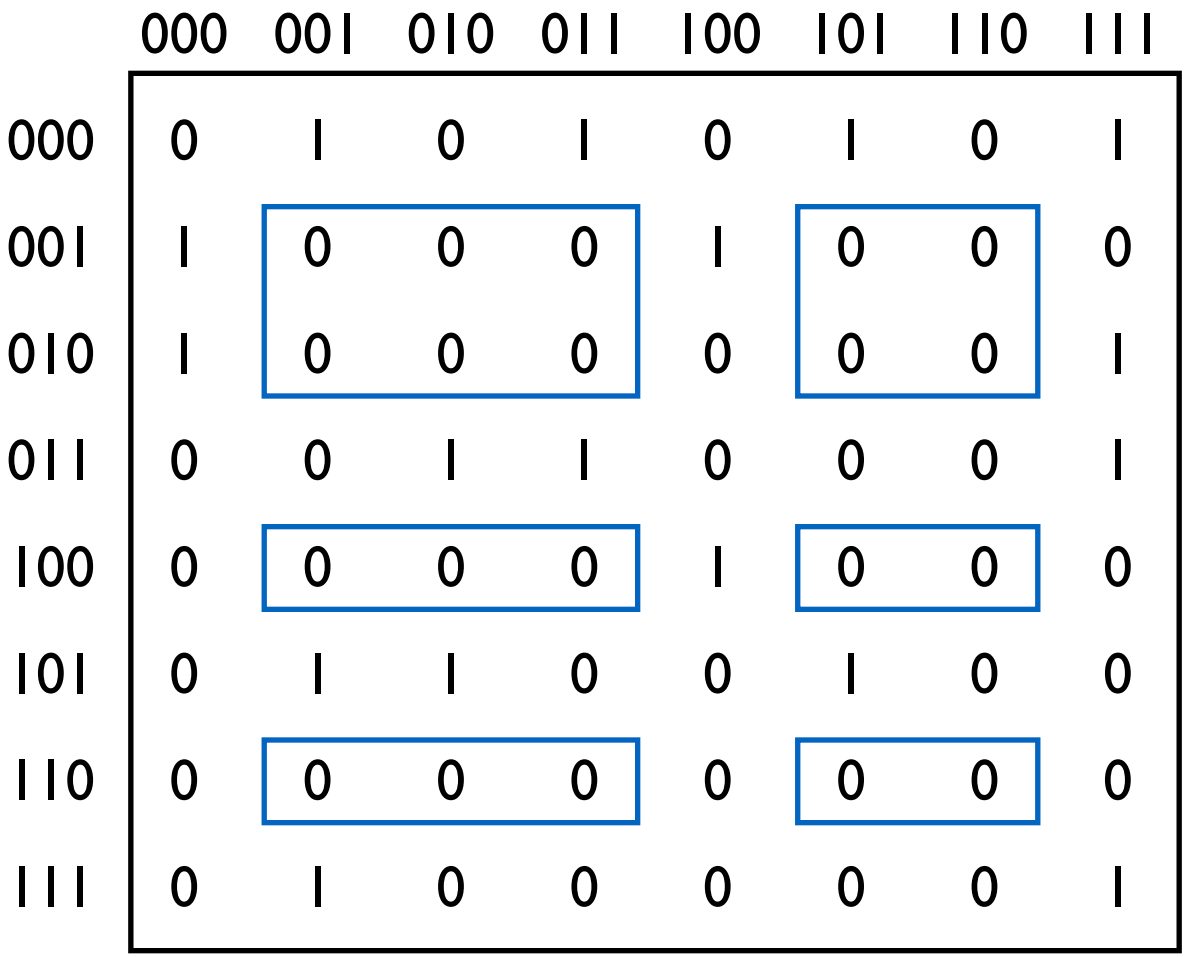
n = 3

x \ y	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

2^n by 2^n matrix

The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

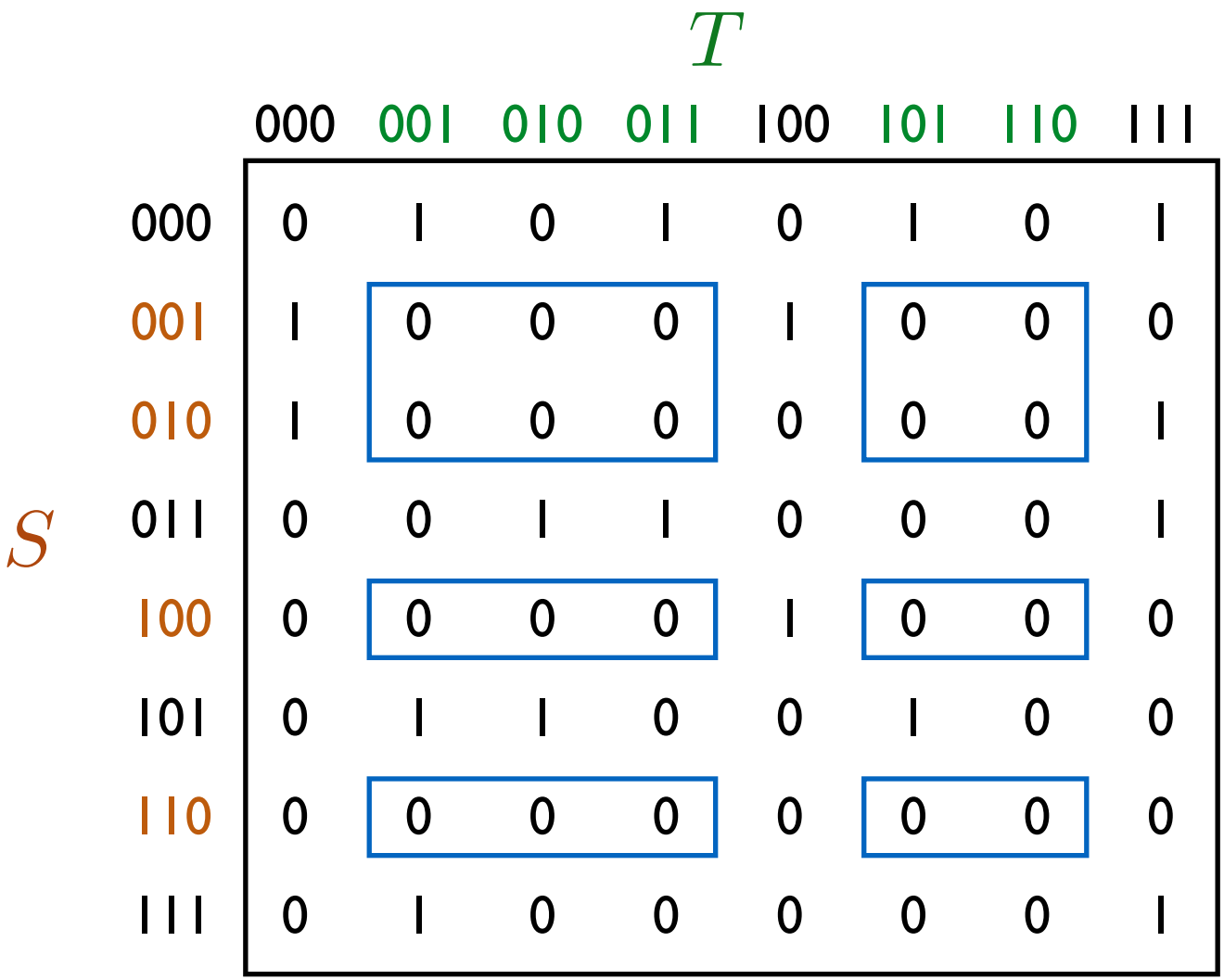


	000	001	010	011	100	101	110	111
000	0	1	0	1	0	1	0	1
001	1	0	0	0	1	0	0	0
010	1	0	0	0	0	0	0	1
011	0	0	1	1	0	0	0	1
100	0	0	0	0	1	0	0	0
101	0	1	1	0	0	1	0	0
110	0	0	0	0	0	0	0	0
111	0	1	0	0	0	0	0	1

A rectangle is of the form $S \times T$ for $S, T \subseteq \{0,1\}^n$

The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

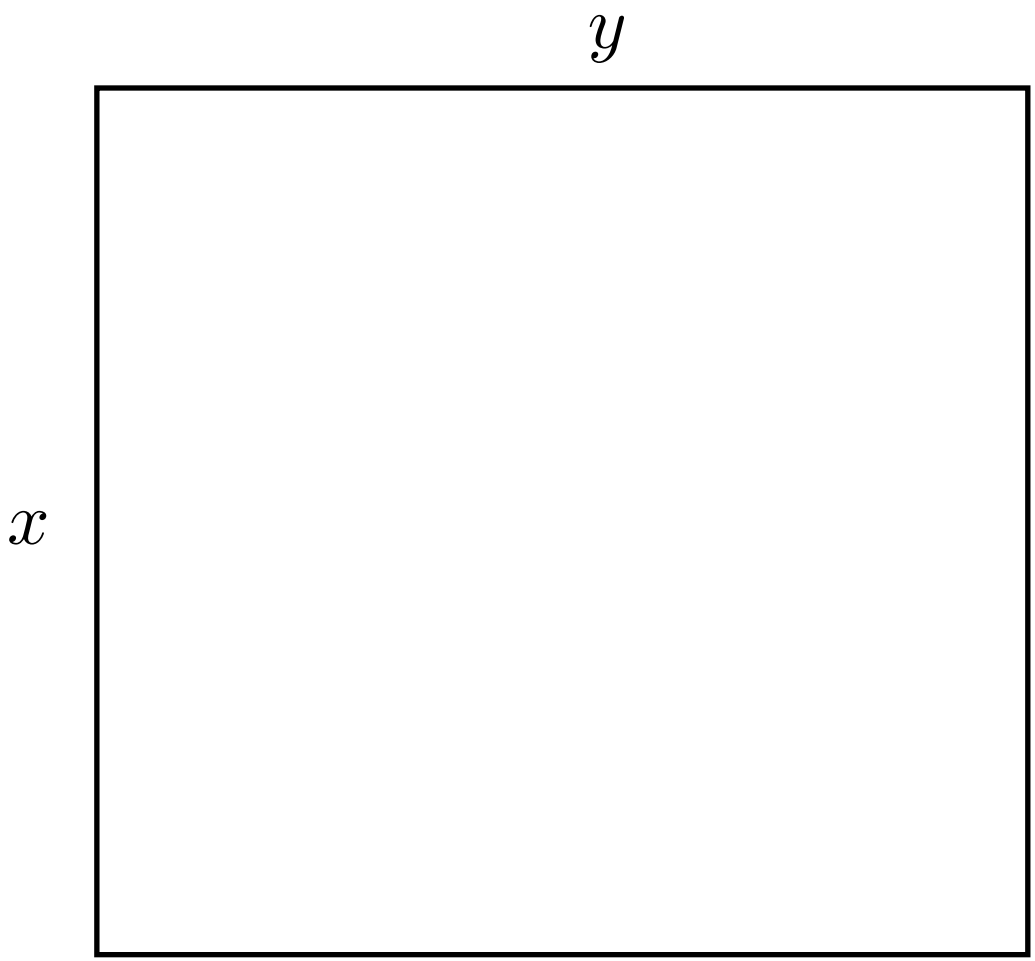


A rectangle is of the form $S \times T$ for $S, T \subseteq \{0,1\}^n$

The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

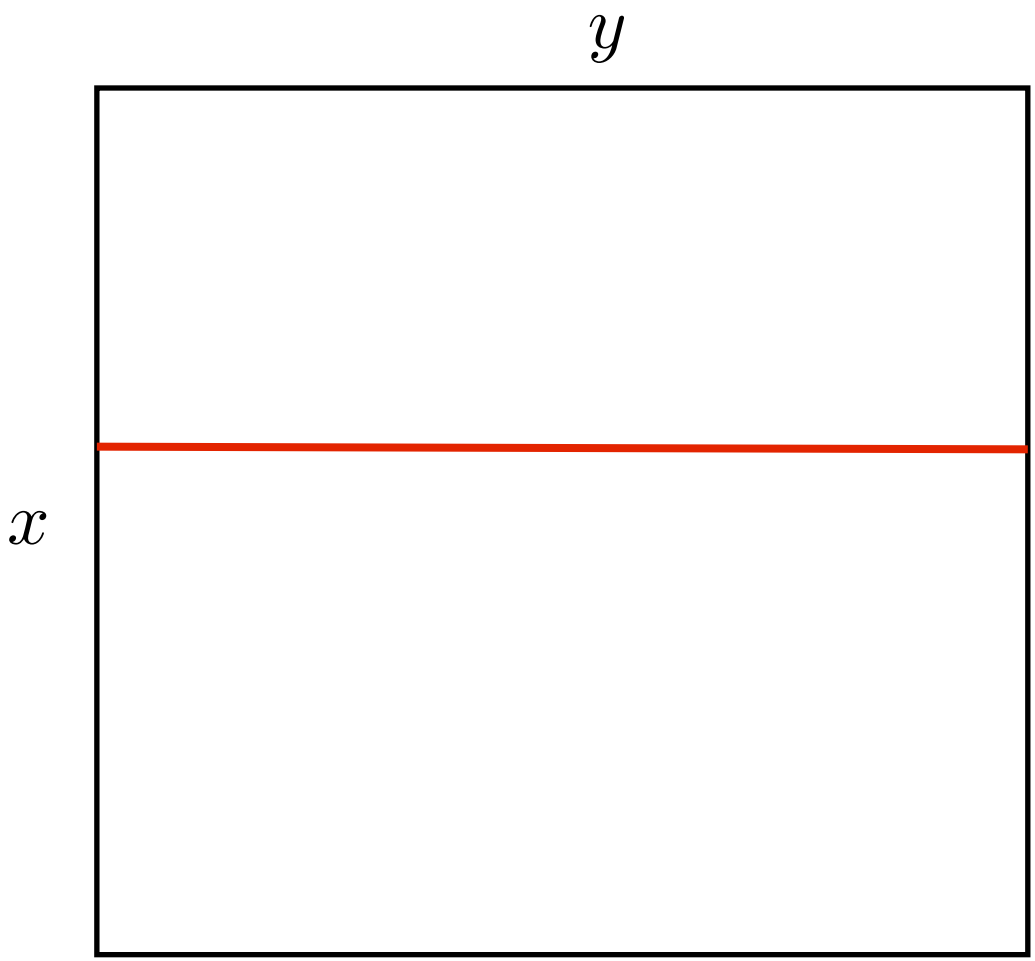
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

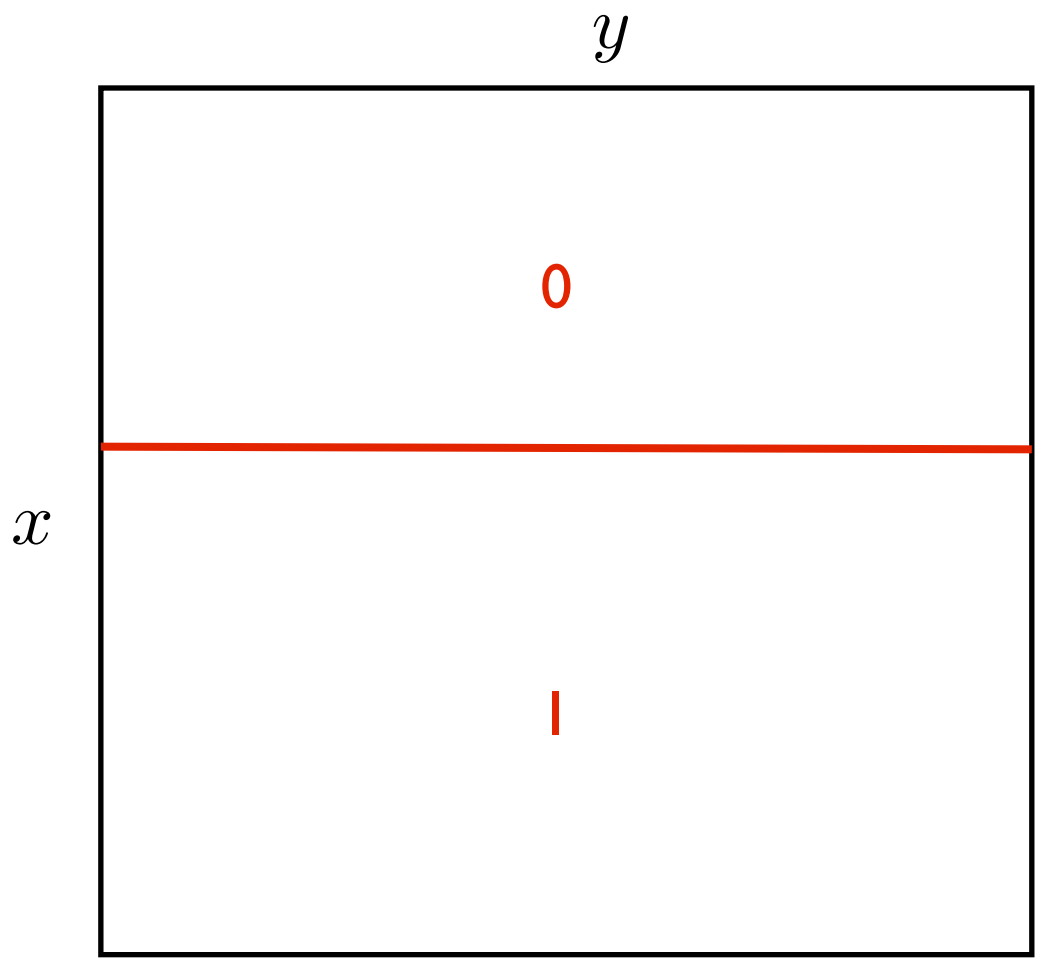
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

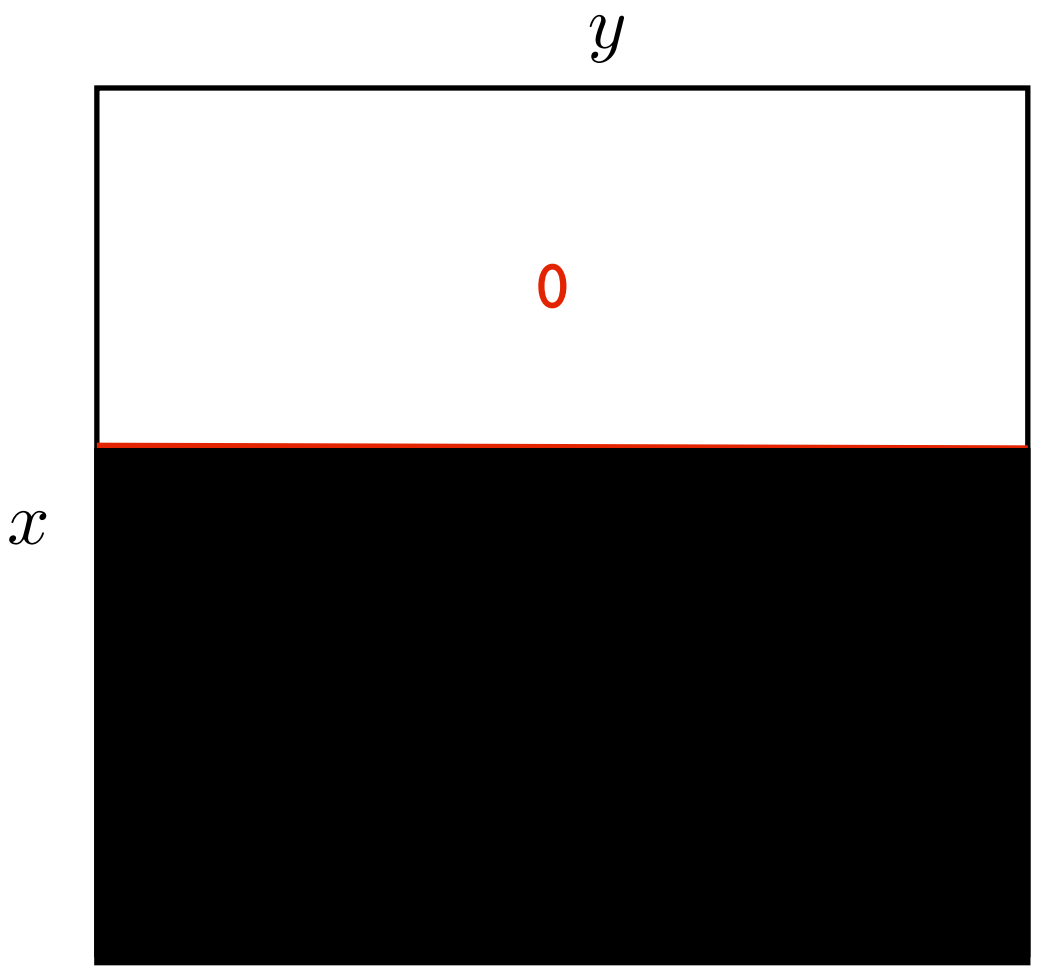
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

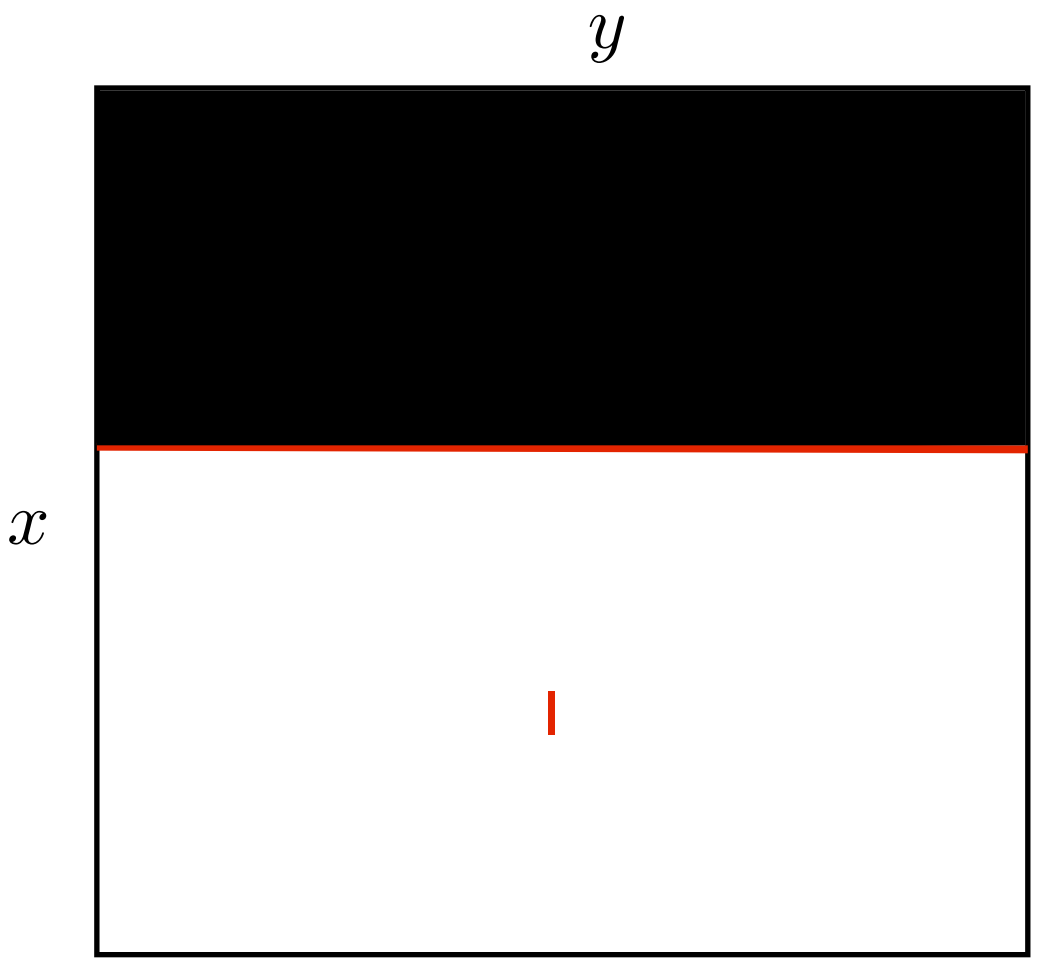
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

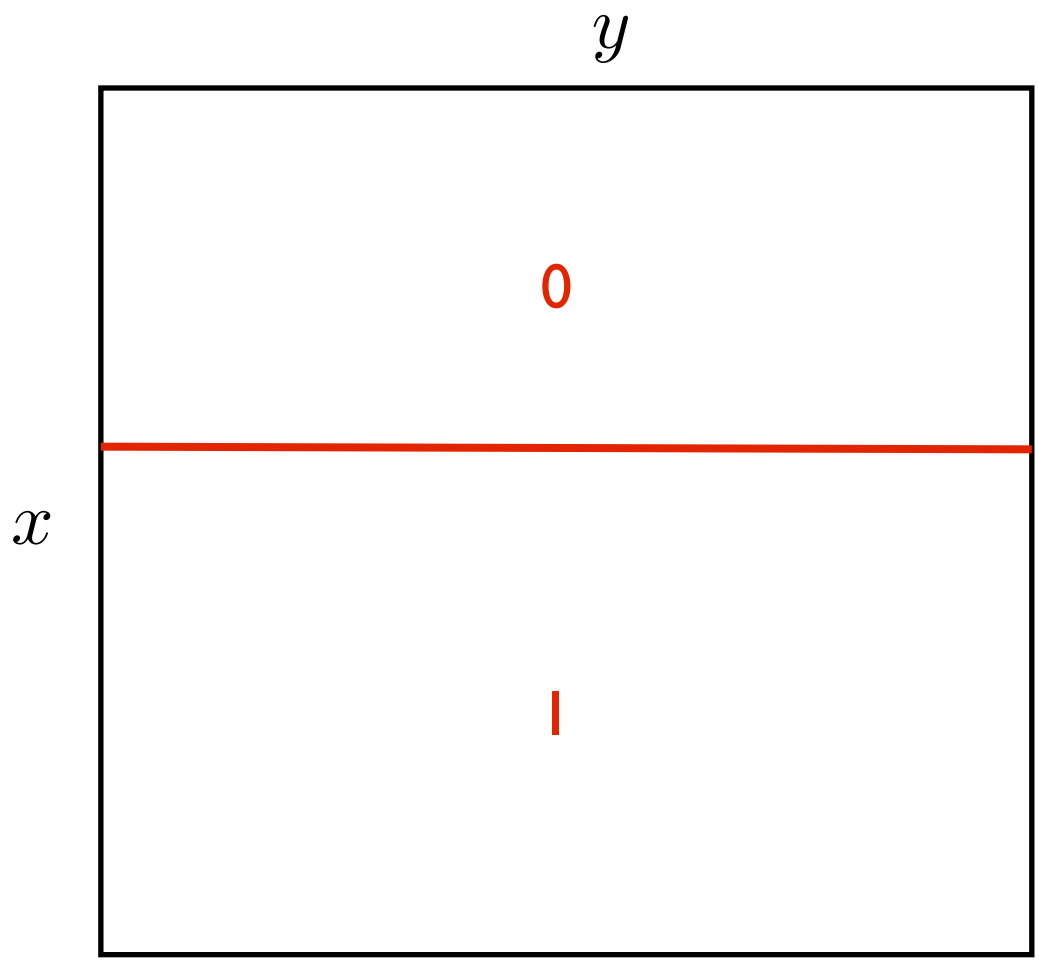
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

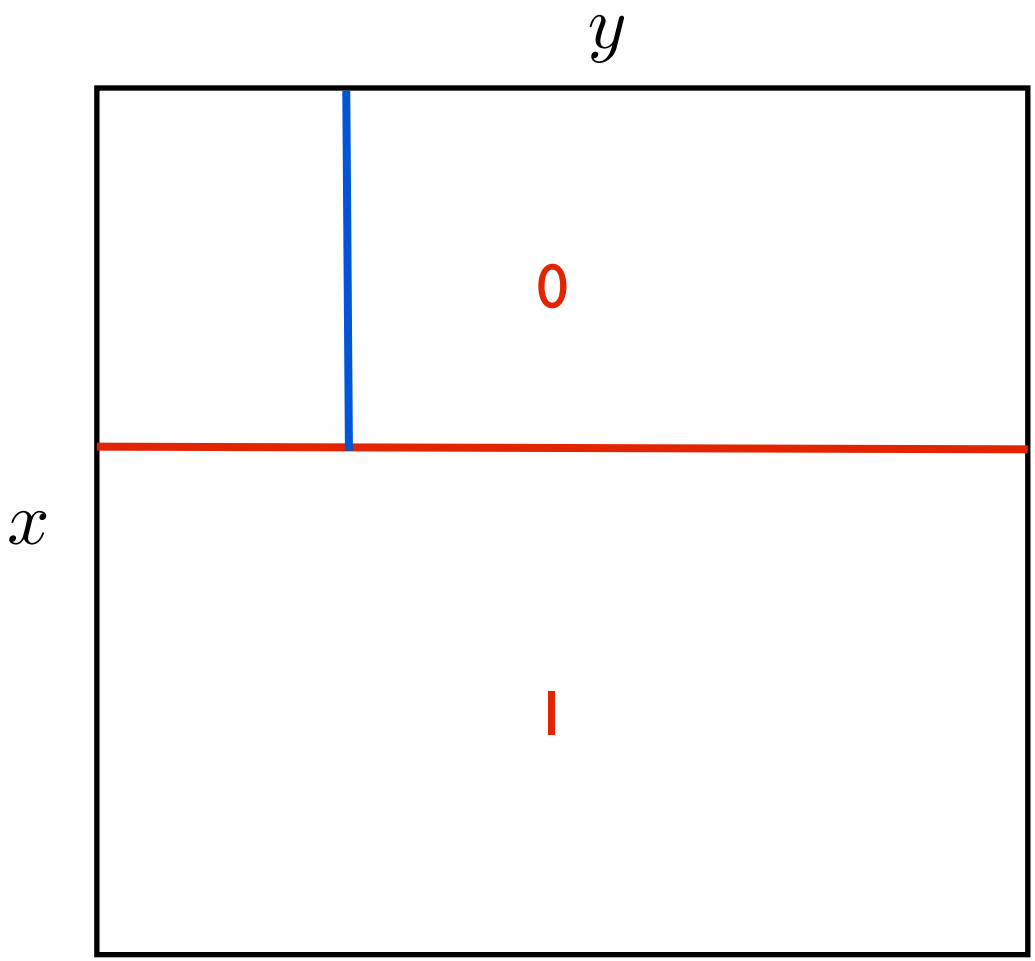
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

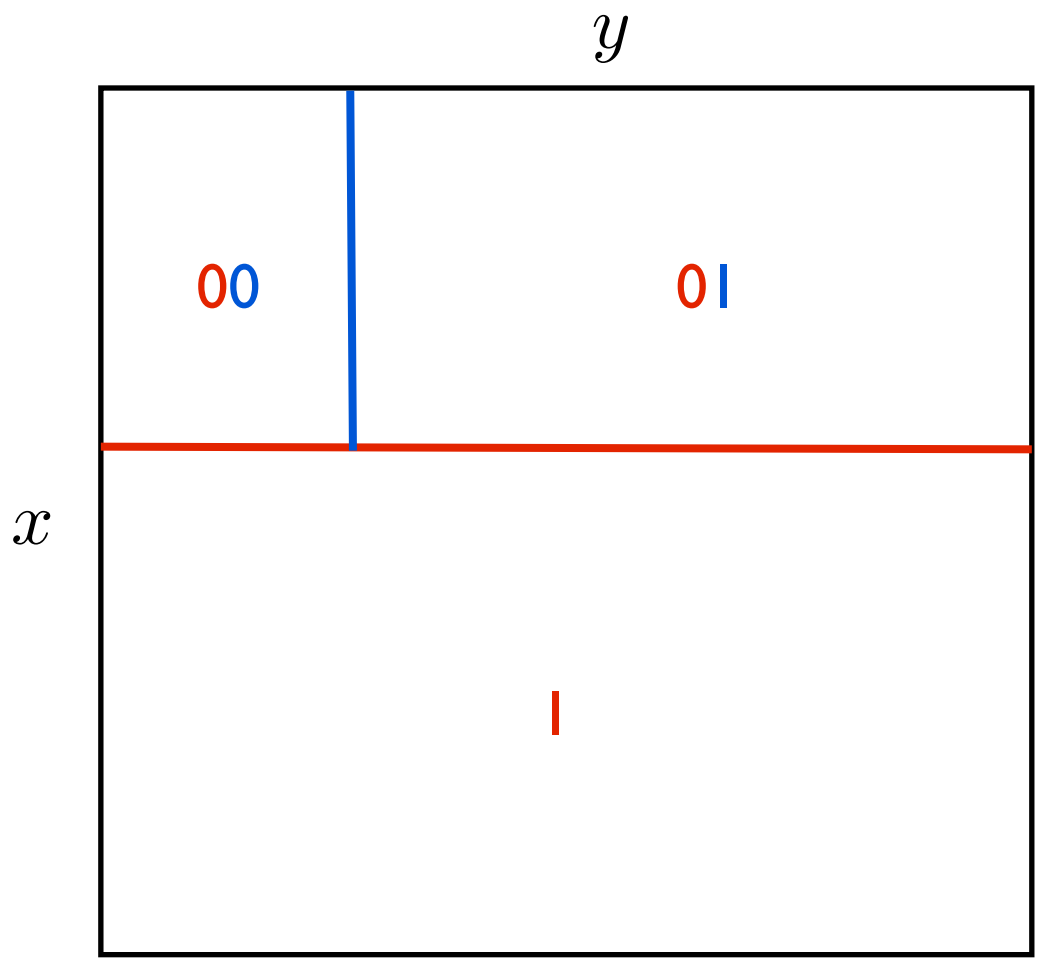
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

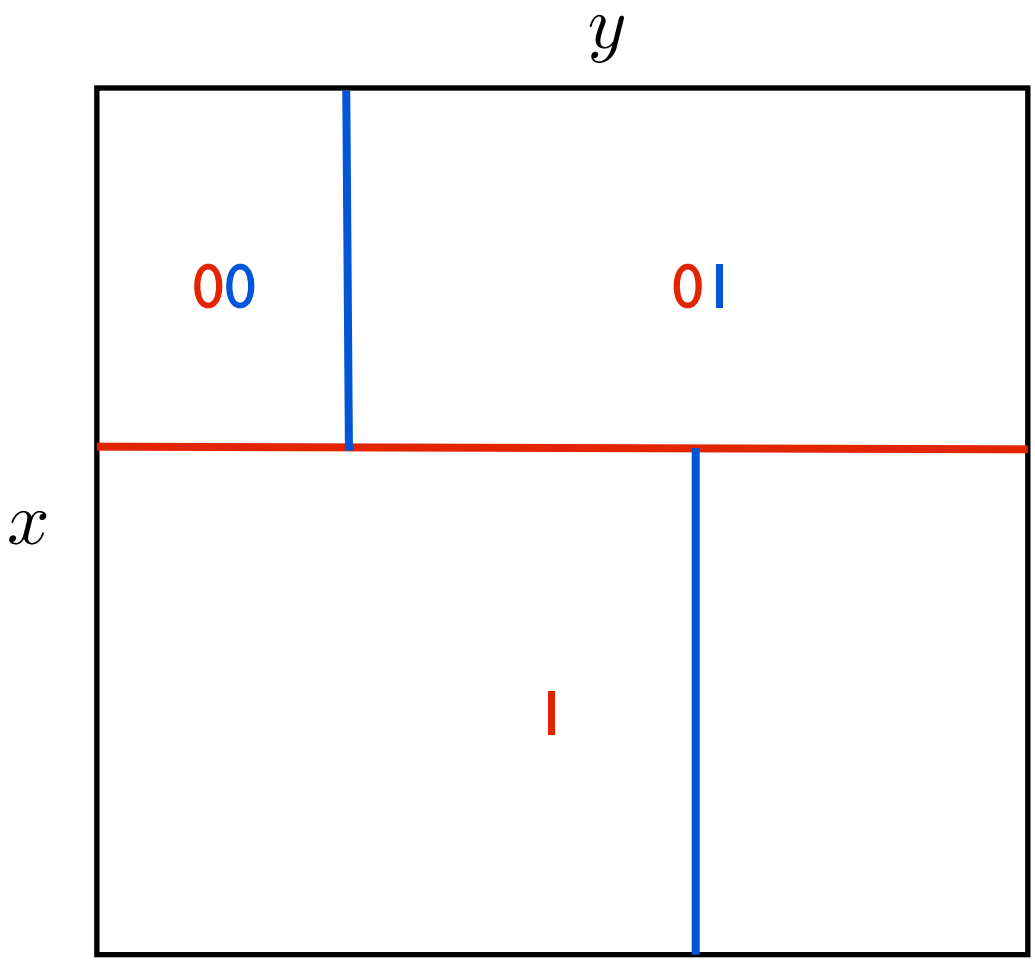
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

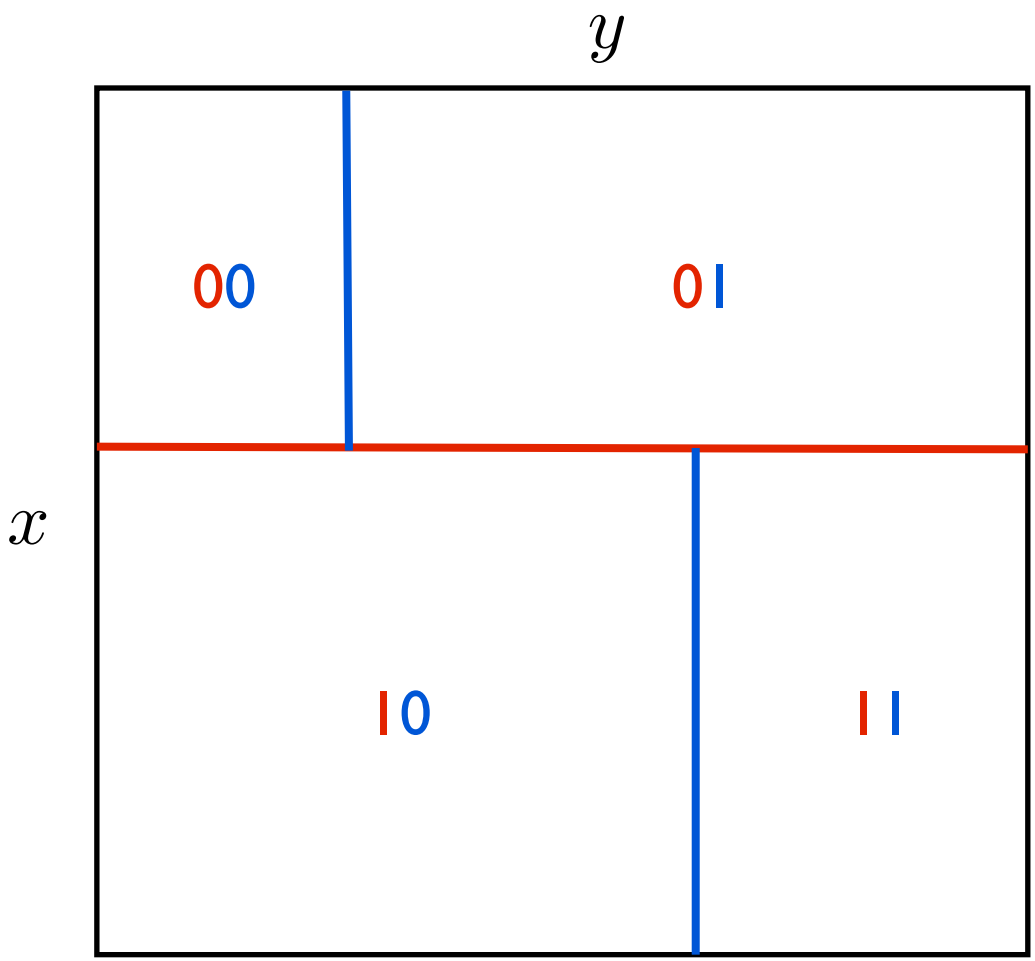
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

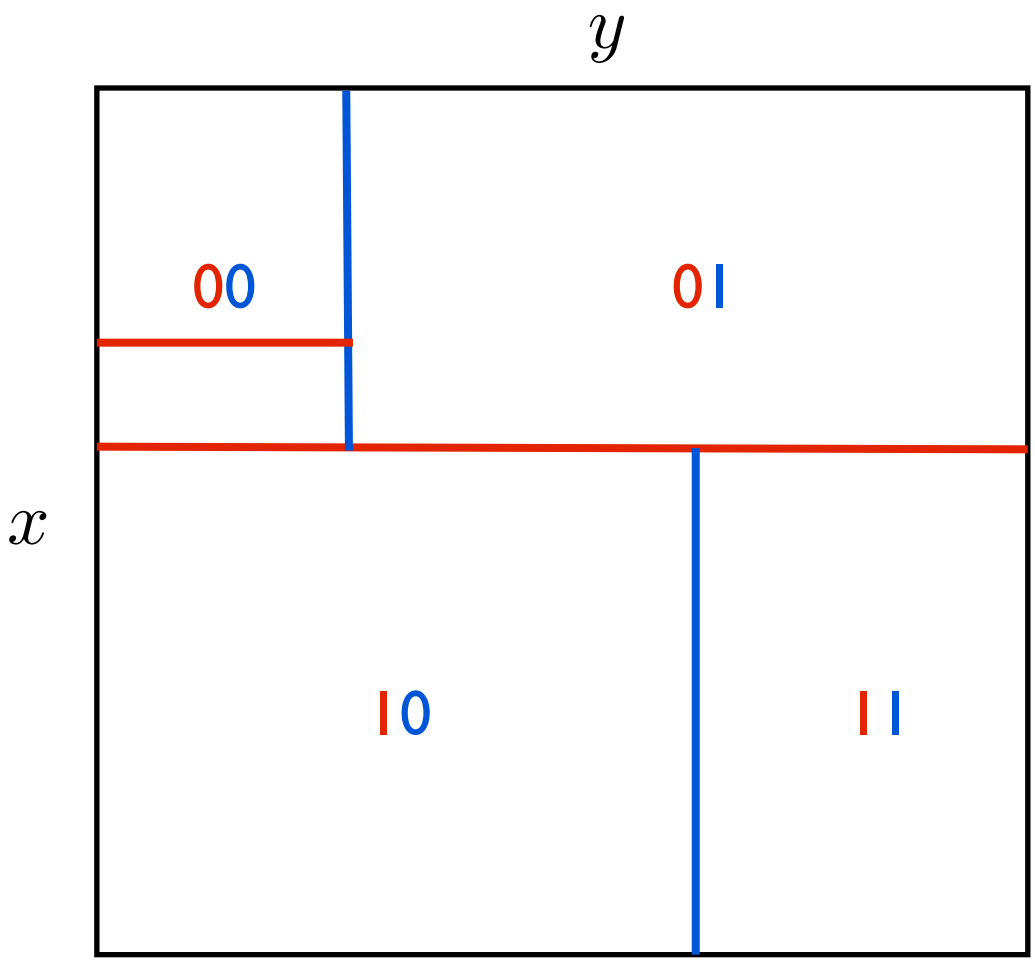
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

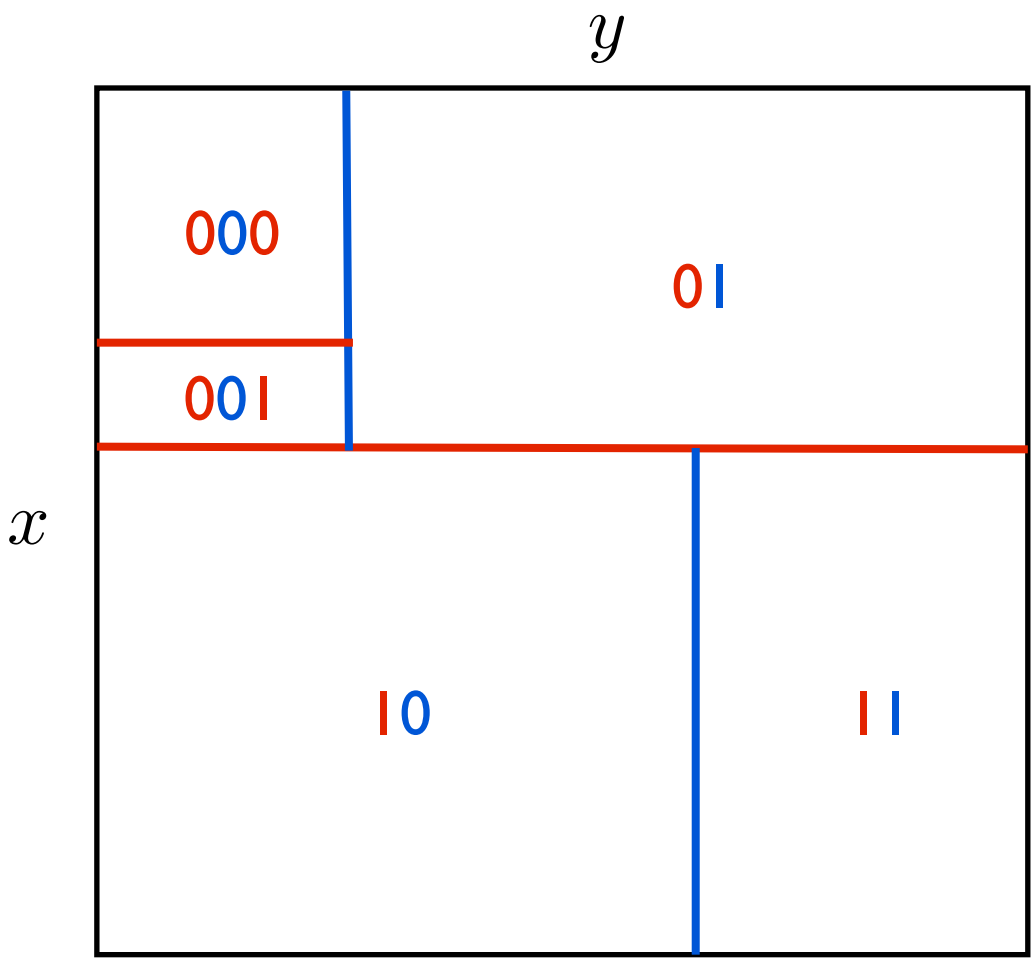
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

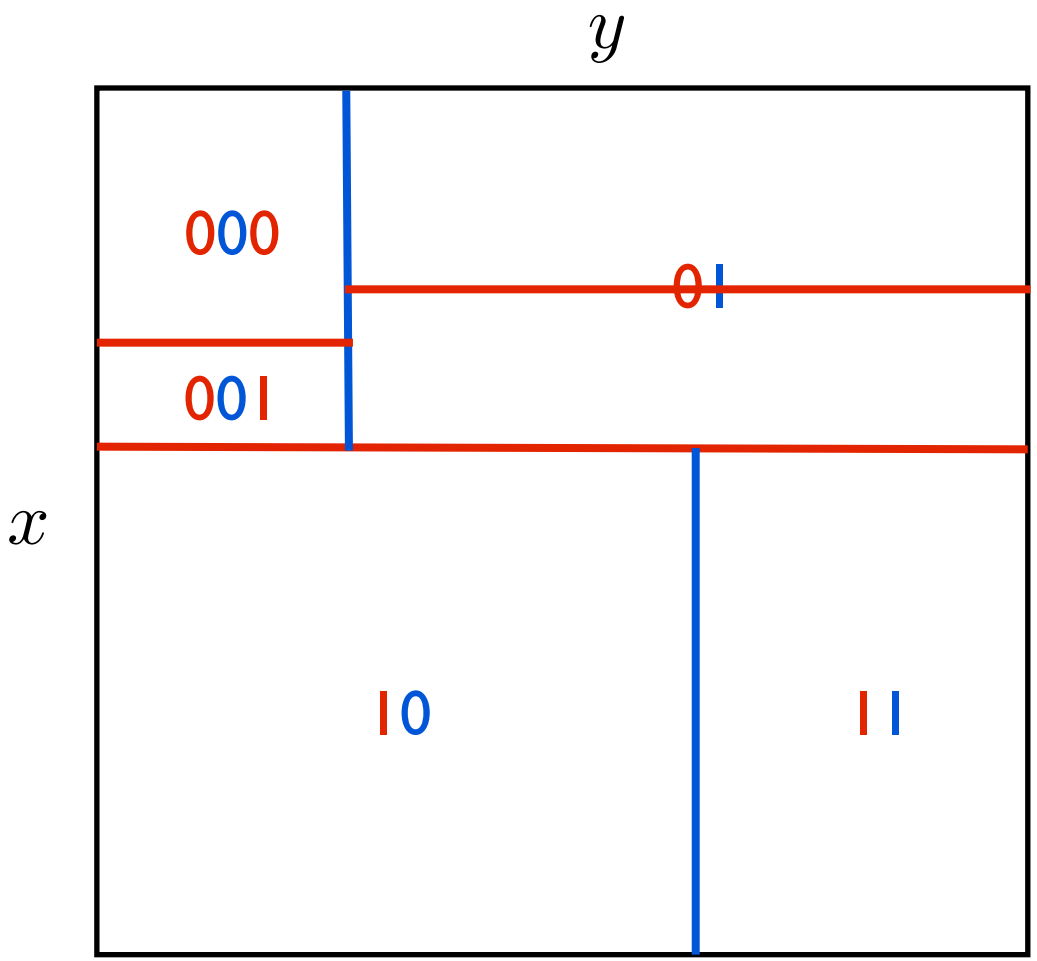
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

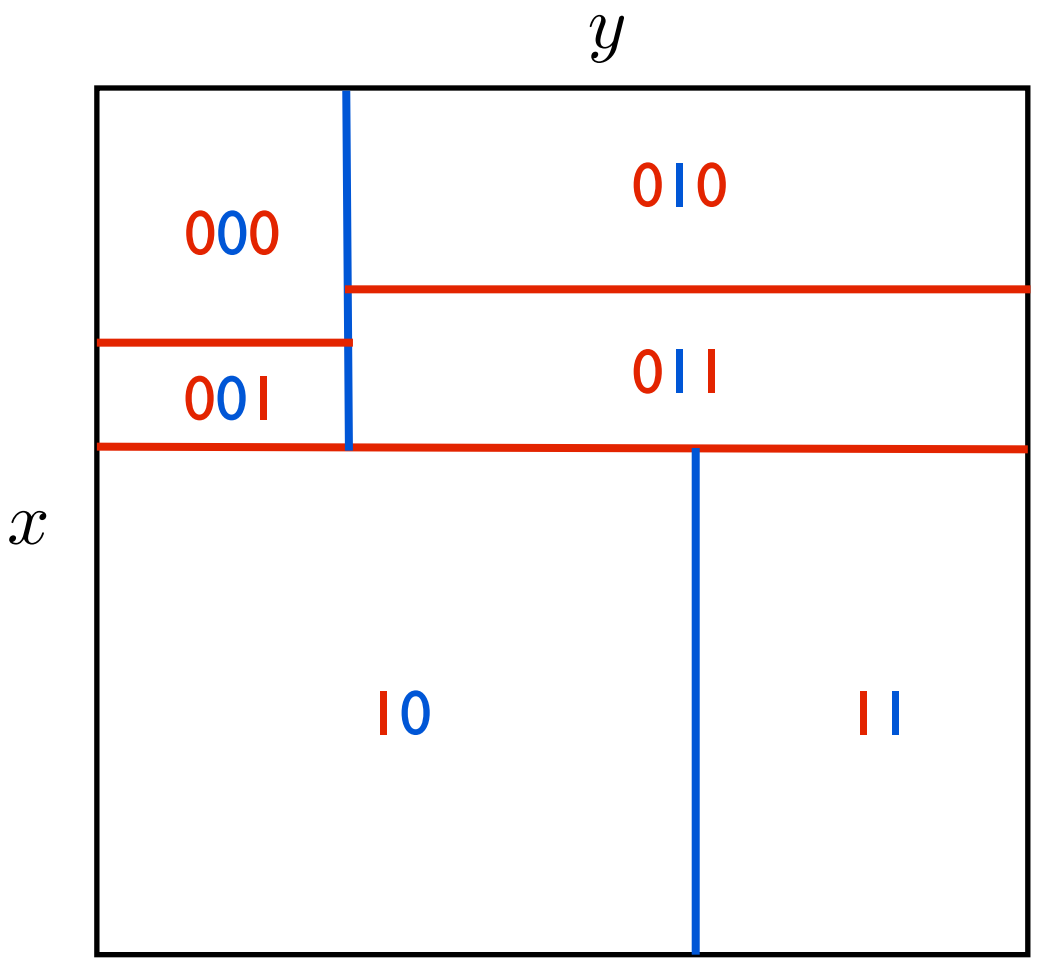
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

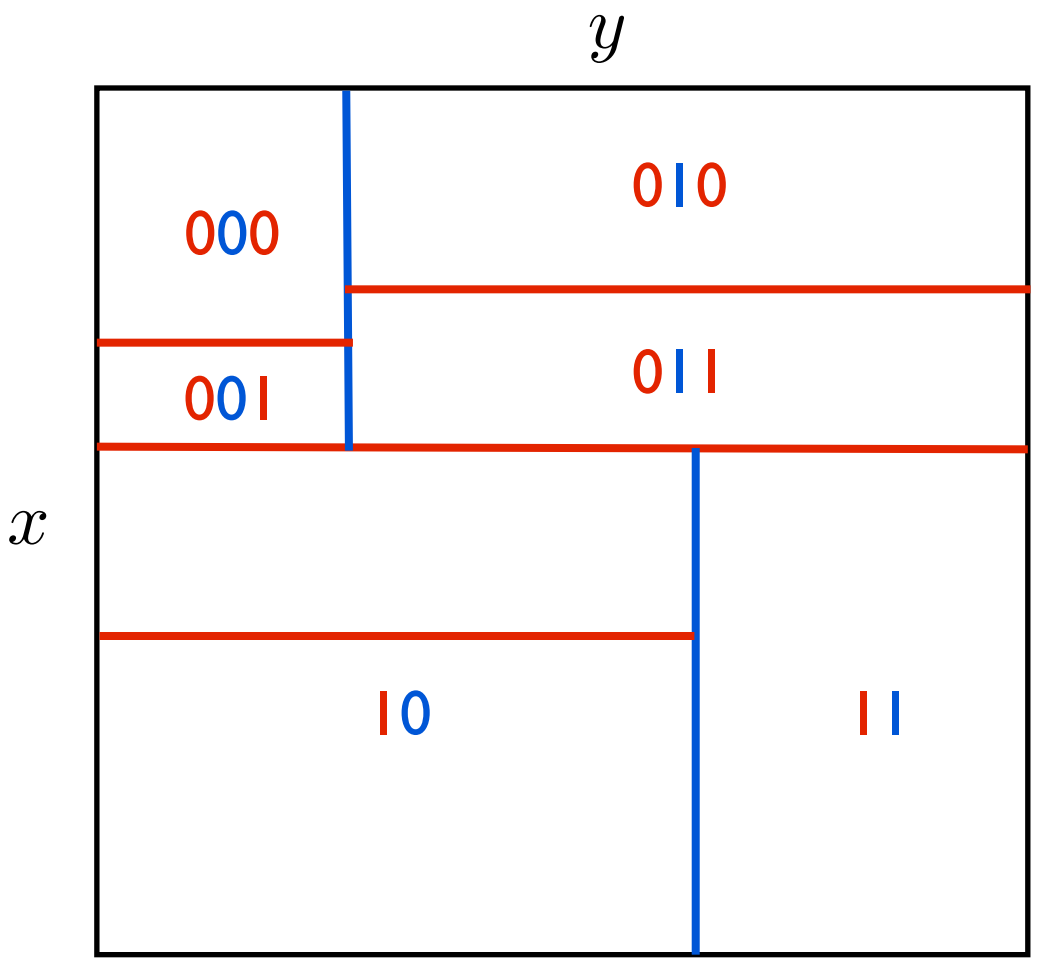
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

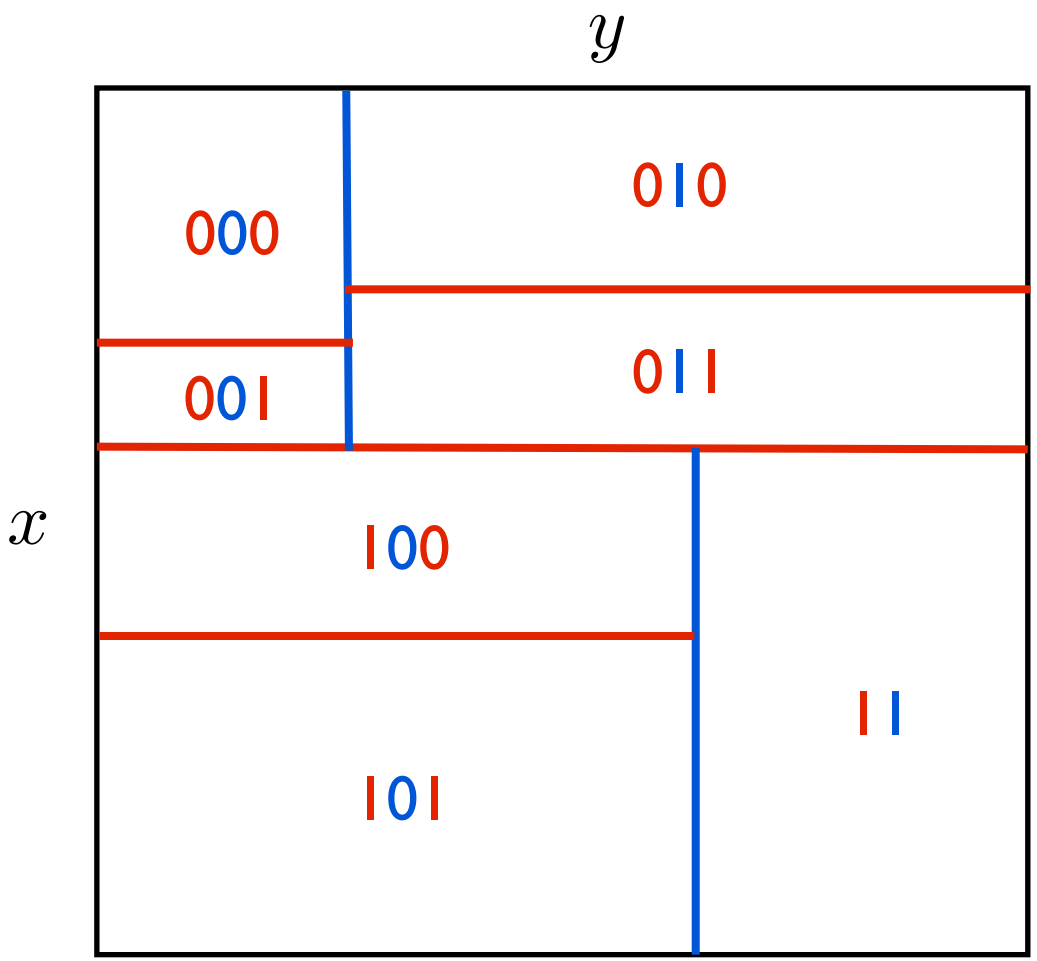
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

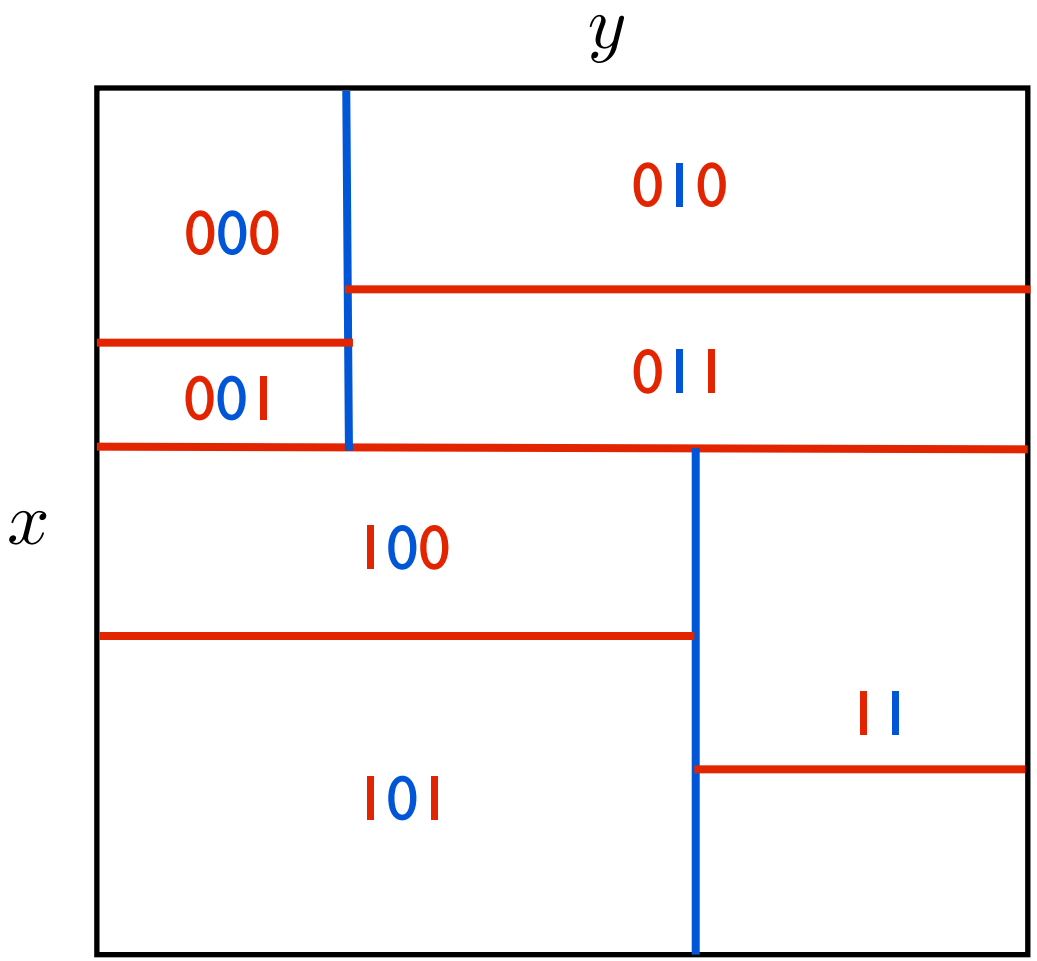
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

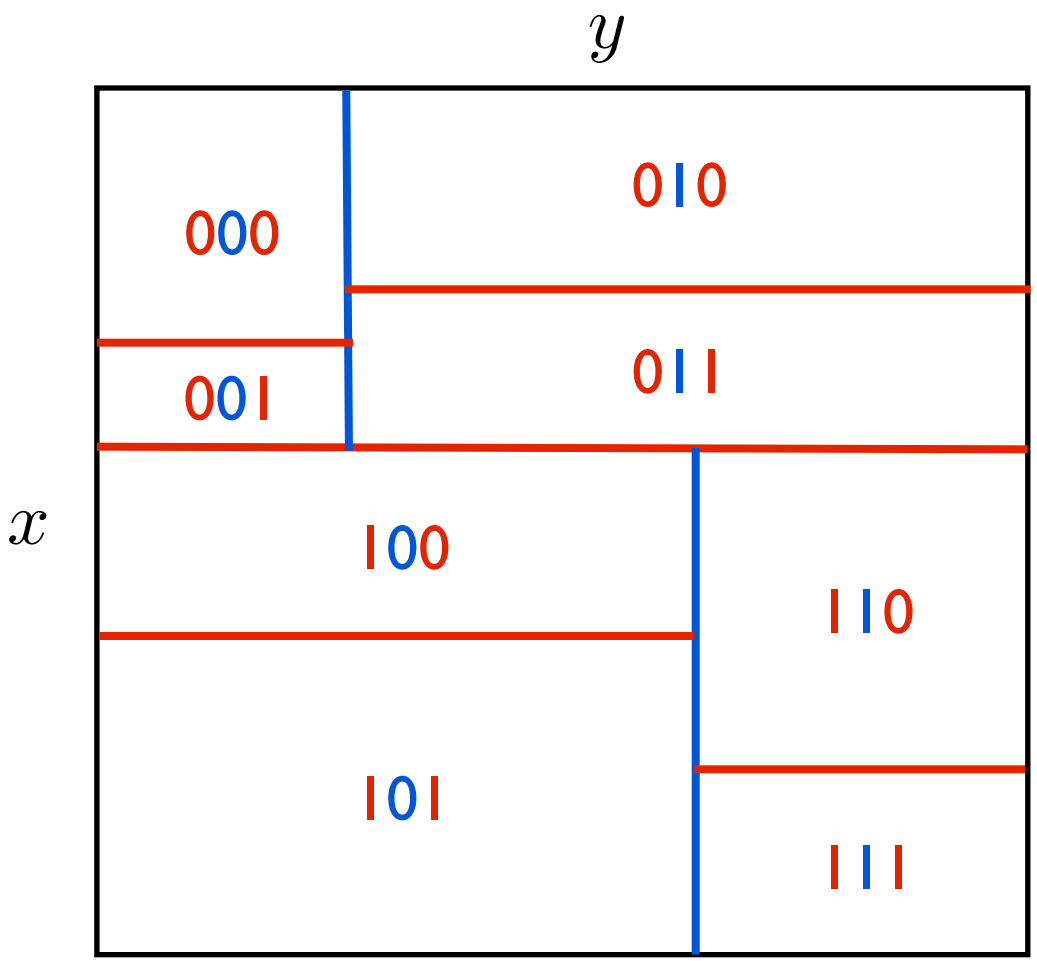
(assume each player sends one bit per round)



The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

(assume each player sends one bit per round)



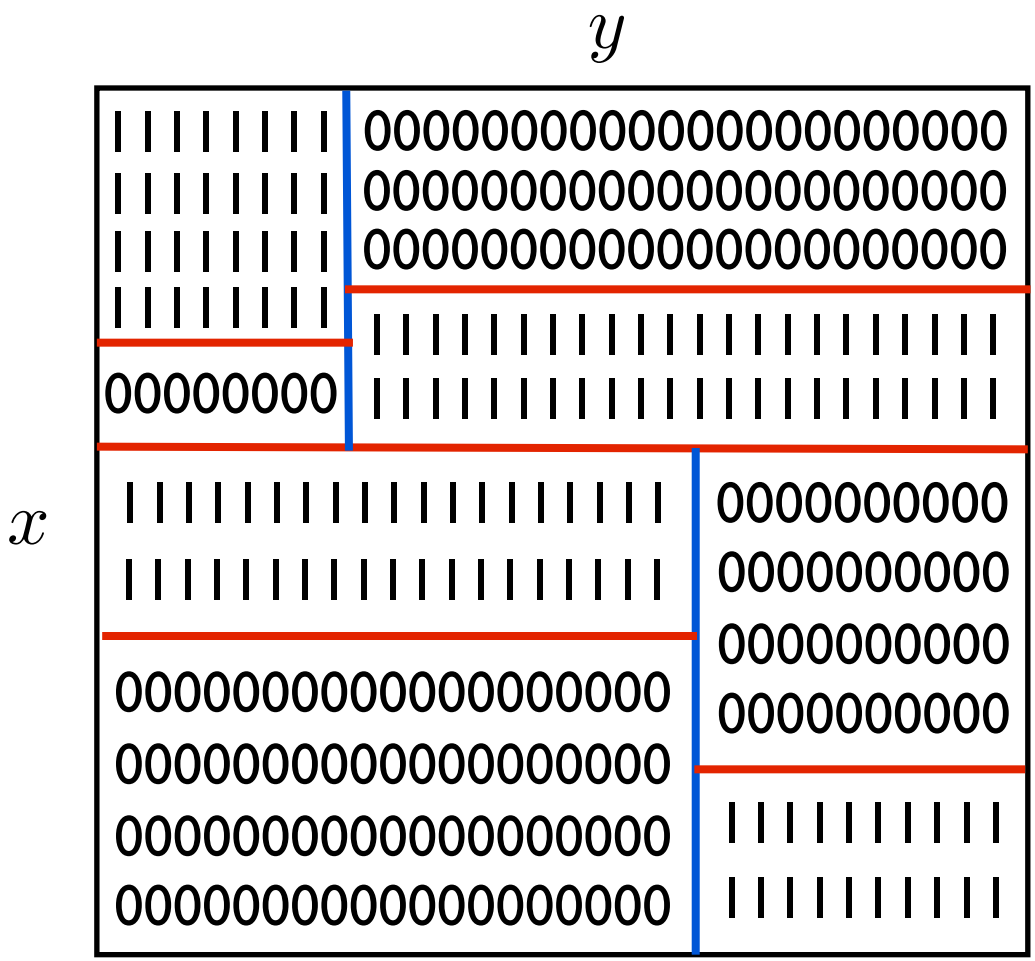
Output of protocol is determined by the communicated bits.

Suppose the protocol is done here. What do we know?

The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

(assume each player sends one bit per round)



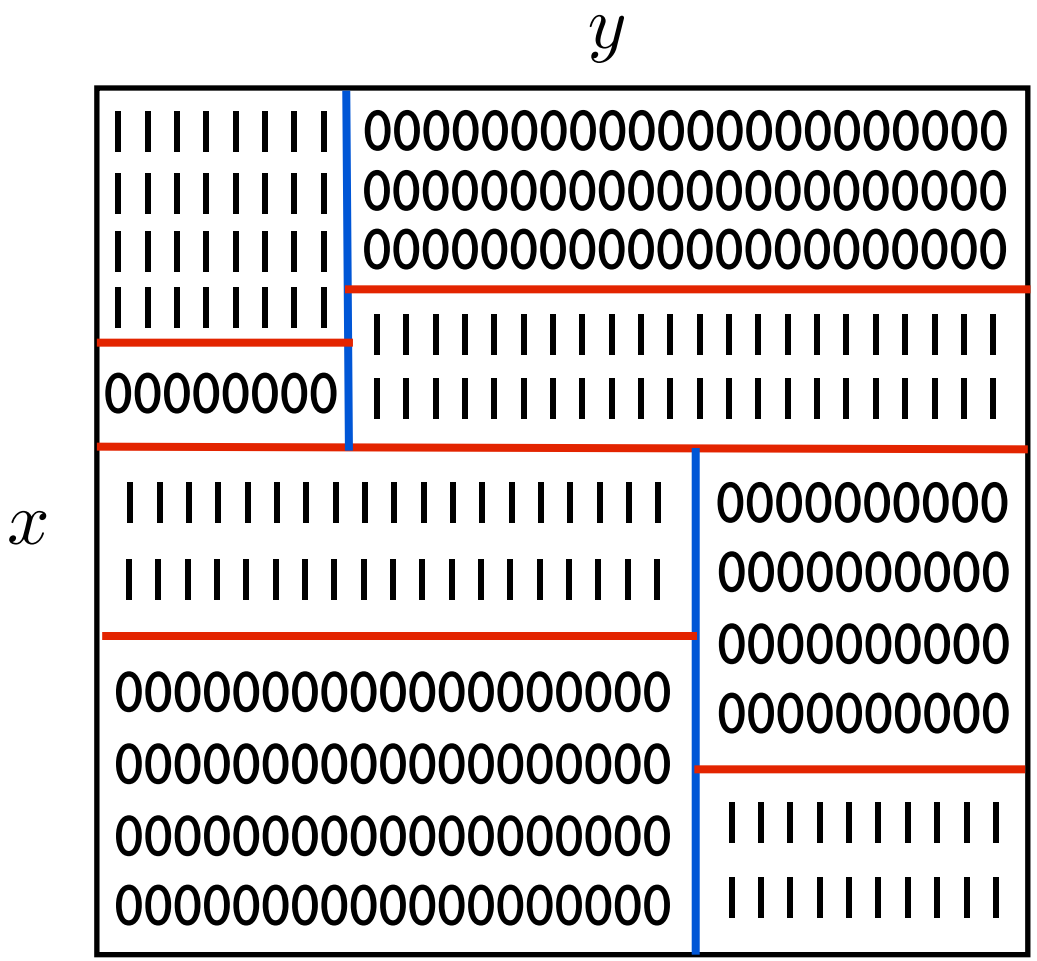
Output of protocol is determined by the communicated bits.

Suppose the protocol is done here. What do we know?

The most important property of a protocol

A protocol partitions *function matrix* into monochromatic rectangles

(assume each player sends one bit per round)



Output of protocol is determined by the communicated bits.

Suppose the protocol is done here. What do we know?

The most important property of a protocol

$$PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$

Protocol:

Alice sends the parity of her input bits.

Bob sends the output of the function.

The most important property of a protocol

$$PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$

	000	011	110	101	111	100	010	001
000	0	0	0	0	1	1	1	1
011	0	0	0	0	1	1	1	1
110	0	0	0	0	1	1	1	1
101	0	0	0	0	1	1	1	1
111	1	1	1	1	0	0	0	0
100	1	1	1	1	0	0	0	0
010	1	1	1	1	0	0	0	0
001	1	1	1	1	0	0	0	0

The most important property of a protocol

$$PAR(x,y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$

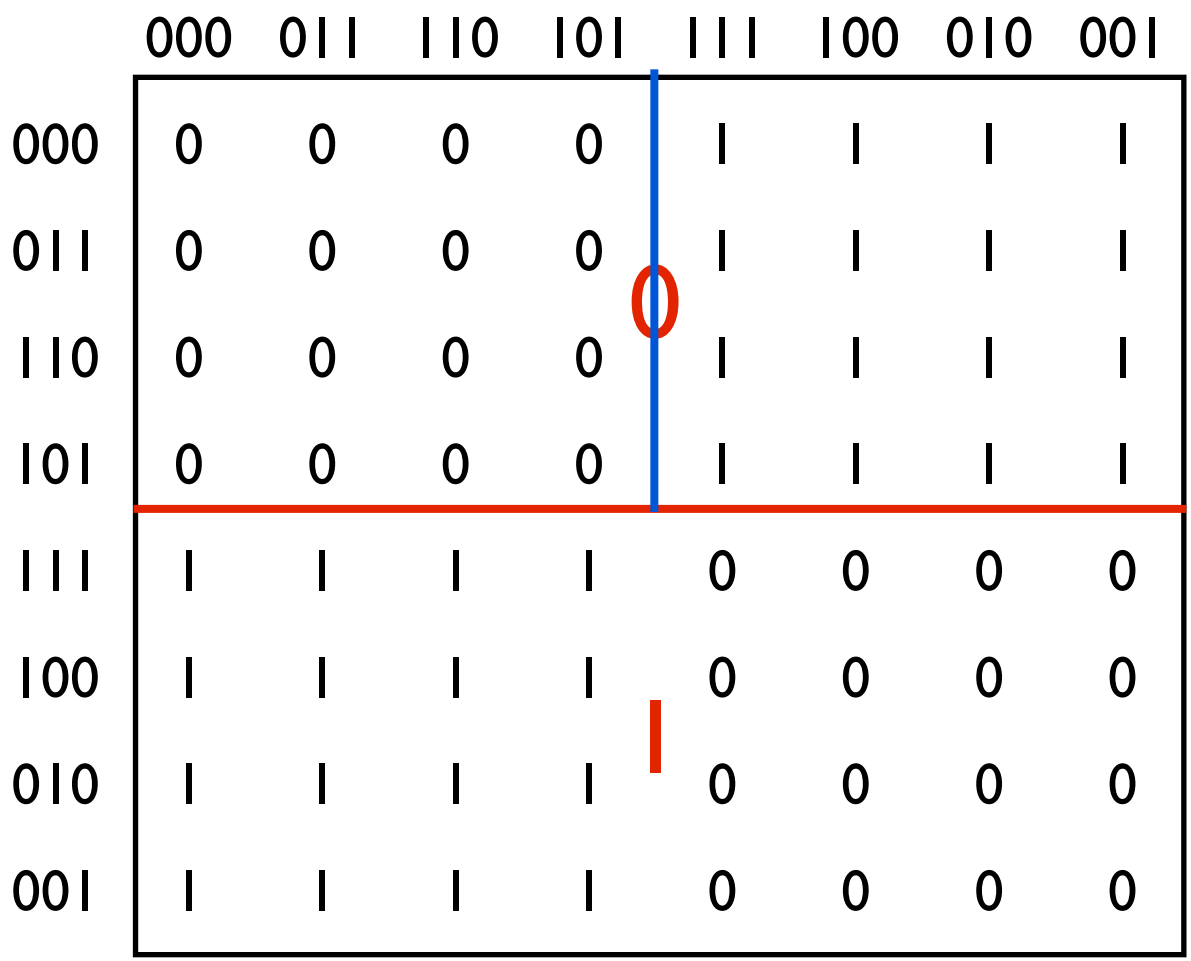
	000	011	110	101	111	100	010	001
000	0	0	0	0	1	1	1	1
011	0	0	0	0	1	1	1	1
110	0	0	0	0	1	1	1	1
101	0	0	0	0	1	1	1	1
111	1	1	1	1	0	0	0	0
100	1	1	1	1	0	0	0	0
010	1	1	1	1	0	0	0	0
001	1	1	1	1	0	0	0	0

0

1

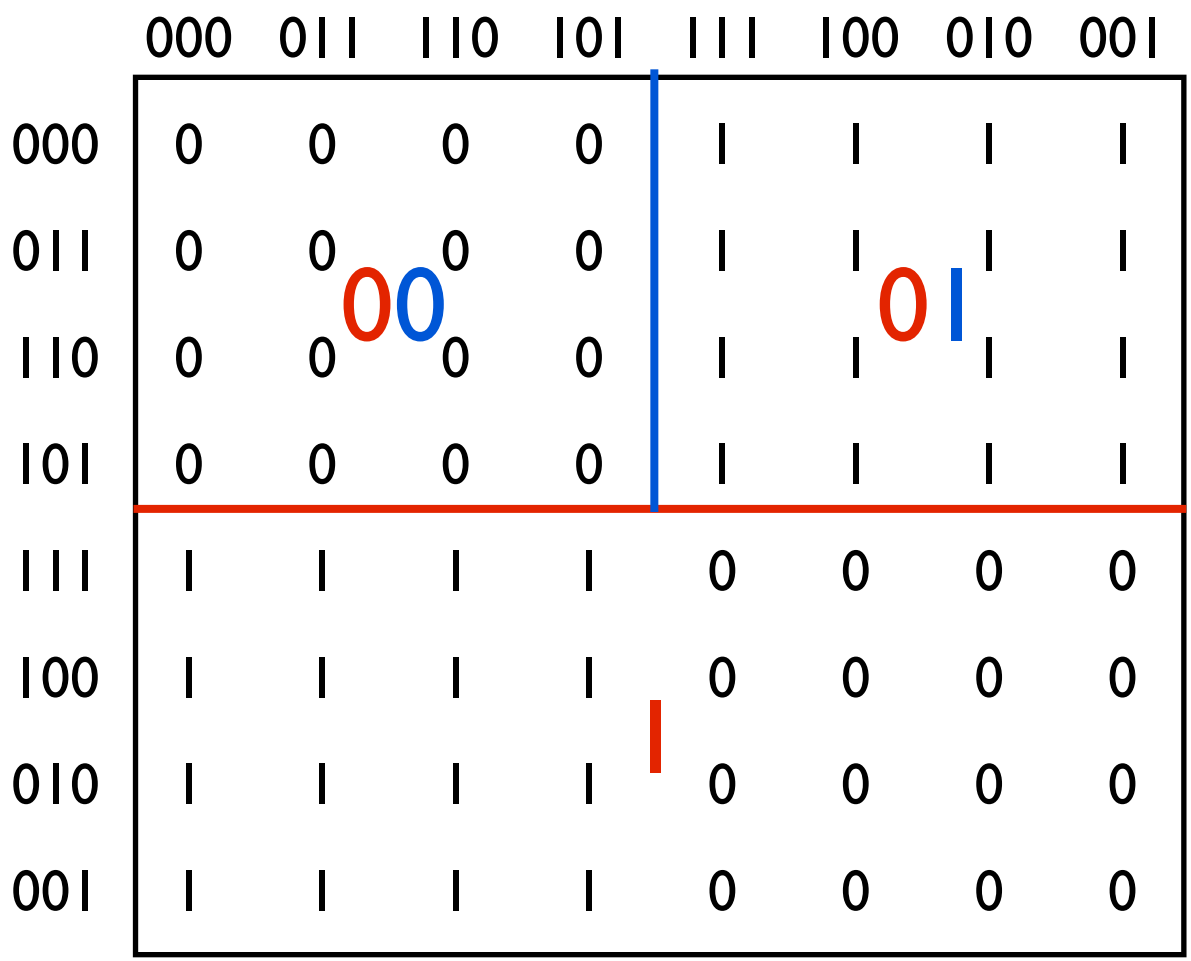
The most important property of a protocol

$$PAR(x,y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$



The most important property of a protocol

$$PAR(x,y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$



The most important property of a protocol

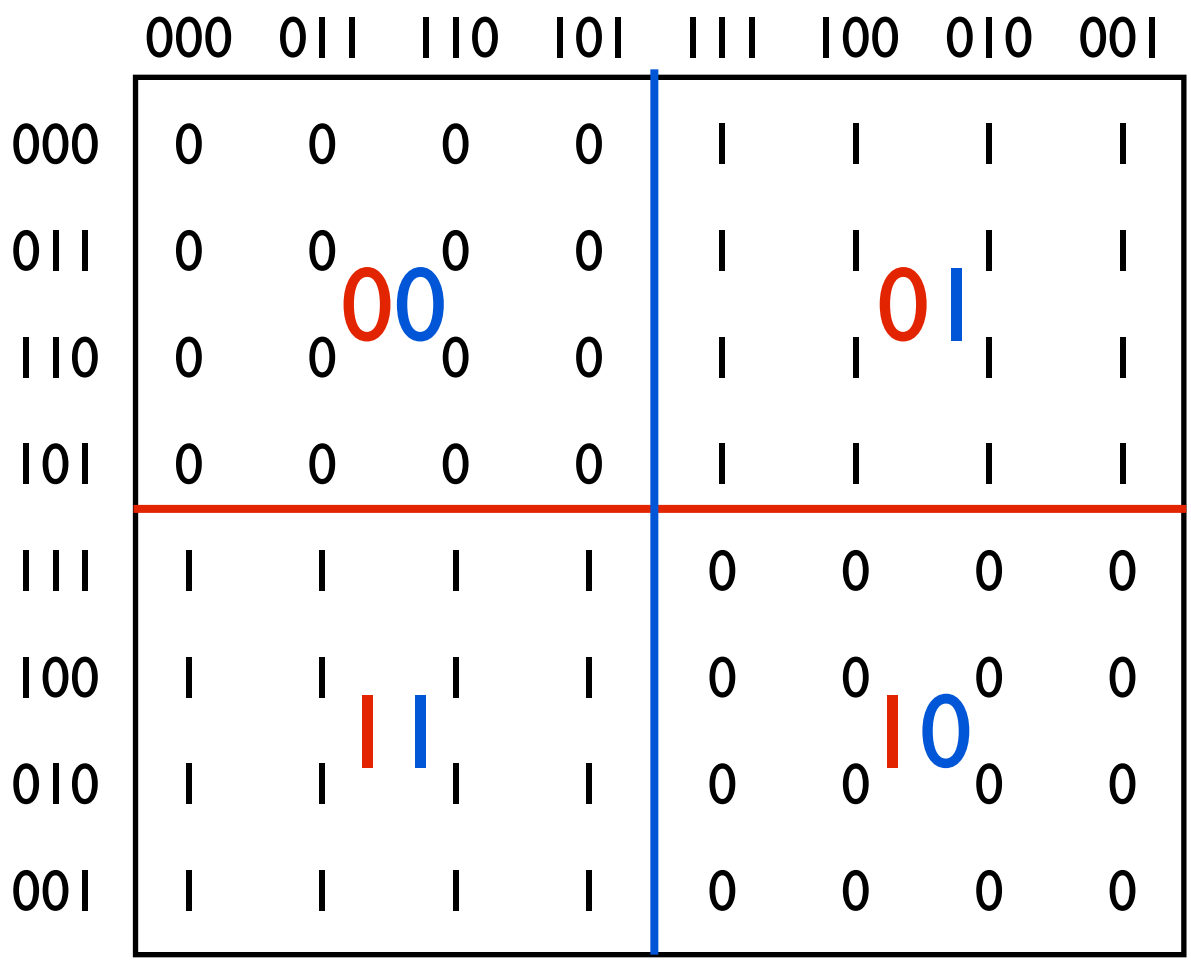
$$PAR(x,y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$

	000	011	110	101	111	100	010	001
000	0	0	0	0	1	1	1	1
011	0	0	0	0	1	1	1	1
110	0	0	0	0	1	1	1	1
101	0	0	0	0	1	1	1	1
111	1	1	1	1	0	0	0	0
100	1	1	1	1	0	0	0	0
010	1	1	1	1	0	0	0	0
001	1	1	1	1	0	0	0	0

00 01

The most important property of a protocol

$$PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod 2$$



The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

Protocol: (a very naive one!!!)

For each $z \in \{0,1\}^n$:

Alice sends 1 if $x = z$, and sends 0 otherwise.

Bob sends 1 if $y = z$, and sends 0 otherwise.

If both players sent a 1 in this round, output 1.

If both players sent different bits, output 0.

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	0	1	0	0
101	0	0	0	0	0	0	1	0
110	0	0	0	0	0	0	0	1
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	0	1	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

repeat same on green sub matrix

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

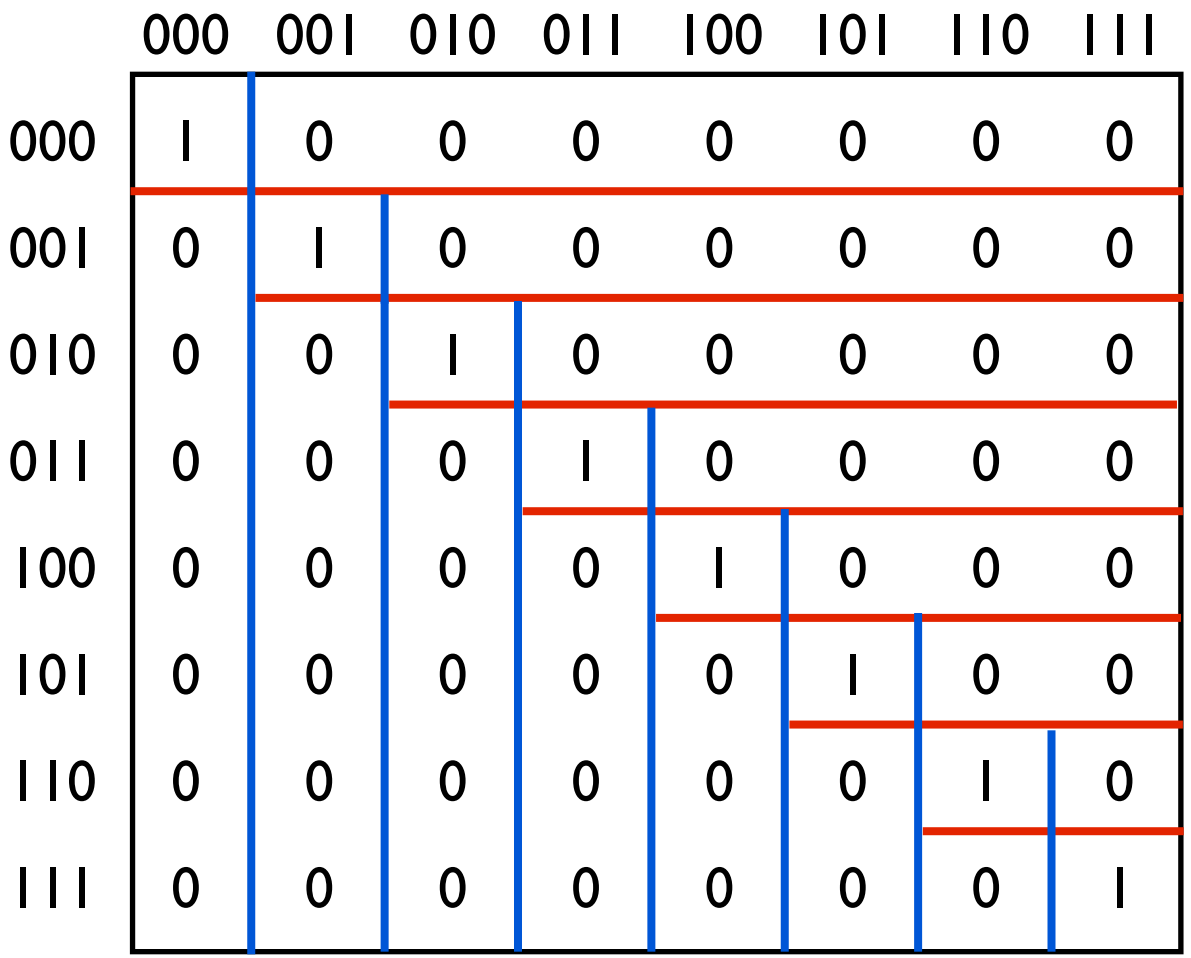
The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

The most important property of a protocol

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$



The most important property of a protocol

Suppose we have a deterministic protocol of cost c that computes a function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.

(every communicated bit at most doubles # rectangles.)

Lower bound for Equality function

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.

	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

Any protocol computing EQ must cover the 1's with monochromatic rectangles.

Lower bound for Equality function

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.

	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

Any protocol computing EQ must cover the 1's with monochromatic rectangles.

Claim: No two 1's can be in the same monochromatic rectangle

Suppose two 1's are in the same rectangle.

Lower bound for Equality function

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.

	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

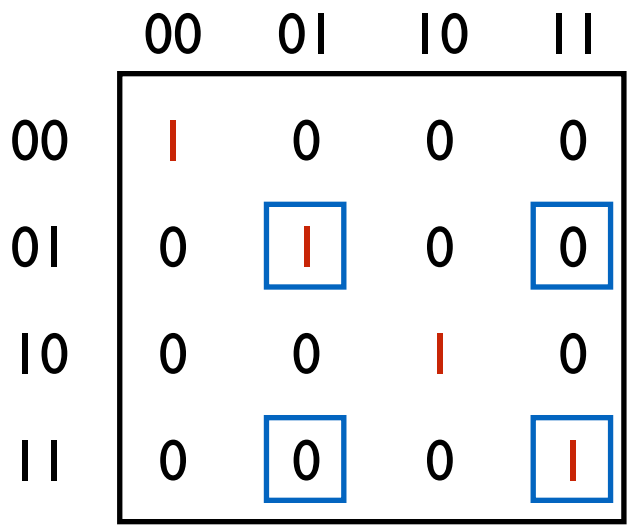
Any protocol computing EQ must cover the 1's with monochromatic rectangles.

Claim: No two 1's can be in the same monochromatic rectangle

Suppose two 1's are in the same rectangle.

Lower bound for Equality function

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.



Any protocol computing EQ must cover the 1's with monochromatic rectangles.

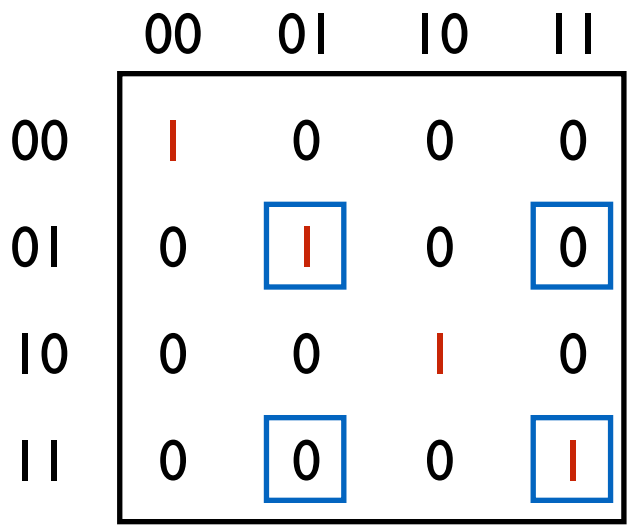
Claim: No two 1's can be in the same monochromatic rectangle

Suppose two 1's are in the same rectangle.

Then there must also be 0's in the rectangle. Contradiction.

Lower bound for Equality function

This protocol partitions the function matrix into at most 2^c monochromatic rectangles.



Any protocol computing EQ must cover the 1's with monochromatic rectangles.

Claim: No two 1's can be in the same monochromatic rectangle

Conclusion: We need a separate rectangle for each 1.

We need at least 2^n rectangles to cover the 1s. $c \geq n$

A general lower bound strategy

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a communication function.

Let $C^D(F)$ be the minimum number monochromatic rectangles needed to partition the function matrix.

A protocol of cost c that computes F partitions the function matrix into at most 2^c monochromatic rectangles.

In particular, for the best protocol computing F,

$$2^c \geq C^D(F)$$

So:

$$\boxed{\mathbf{D}_2(F) \geq \lceil \log_2 C^D(F) \rceil}$$

A general lower bound strategy

$$\mathbf{D}_2(F) \geq \lceil \log_2 C^D(F) \rceil$$

For $F = EQ$:

We need 2^n rectangles to cover the 1s.

We need at least one rectangle to cover the 0s.

So $C^D(EQ) \geq 2^n + 1$

Therefore $\mathbf{D}_2(EQ) \geq \lceil \log_2(2^n + 1) \rceil = n + 1$

$$\mathbf{D}_2(EQ) = n + 1$$

A few remarks

$$\mathbf{D}_2(F) \geq \lceil \log_2 C^D(F) \rceil$$

There are some interesting lower bound techniques that lower bound $C^D(F)$.

One notable example: $C^D(F) \geq \mathrm{rank}(F)$

This lower bound technique is pretty good in general:

$$\mathbf{D}_2(F) = O(\log^2 C^D(F))$$

Proving lower bounds for randomized communication complexity of functions is considerably harder.

But still doable!

Take-Home Message

Communication complexity studies natural distributed tasks.

Communication complexity (lower bounds) has **many** interesting applications.

Lower bounds can be proved using a variety of tools:
combinatorial, algebraic, analytic, information theoretic